COMP90082-2024-FL-Koala Home

Supervisor:

Wei Wang

Team member:

ID	Name	Role
1133093	Junhao Kong	DL
1397061	Lingyi Kong	SM
1132416	Yijun Liu	DEL
1367102	Zhihao Liang	QA
1373110	Yuncong Ji	AL
1351342	Lin Duan	PO

Table of Content

Project Overview

The project aims to streamline the management and execution of Python scripts used in radiation oncology at ONJ, a department within Austin Health. These scripts facilitate various tasks related to treating cancer patients using imaging devices and a linear accelerator for radiation delivery. The current process involves manual execution of scripts, posing challenges in version control, accessibility, and maintenance.

To address these issues, the project proposes two main components:

1. **Prototype Web Interface for Script Execution:** Utilizing Flask or a suitable alternative, a web-based dashboard will be developed. This interface will allow users to execute scripts without the need for Python installation. It aims to simplify the execution process, particularly for non-Python users, and enable centralized access to scripts.
2. **GitHub Repository for Version Control:** The project will organize the scripts into a GitHub repository, enabling version control, collaboration, and centralized access. GitHub's features, such as branching, pull requests, and user management, will facilitate efficient script management and development.

Background Description

Goals and Motivation

Personas

DO-BE-FEEL and GOAL MODEL

Background Description

Radiation oncology at ONJ within Austin Health represents a critical intersection of medical expertise and technological innovation in cancer treatment. With the ever-evolving landscape of oncological research and technology, the department continually seeks to optimize its processes to deliver the best possible care to patients.

Central to the treatment process are advanced imaging techniques such as CT and MRI, which provide crucial insights into the location and extent of tumors within patients' bodies. These images serve as the foundation for treatment planning, where precise calculations and simulations are performed to determine the optimal delivery of radiation to target cancerous tissues while minimizing damage to healthy surrounding tissue.

The linchpin in this treatment paradigm is the linear accelerator, a sophisticated piece of machinery that generates high-energy radiation beams used to eradicate cancer cells. The precise calibration and delivery of these beams are guided by complex algorithms and calculations, which are often facilitated by custom Python scripts.

Given the inherently interdisciplinary nature of radiation oncology, the department comprises individuals with diverse skill sets and backgrounds, ranging from medical professionals to technologists and researchers. This diversity presents both challenges and opportunities in terms of streamlining processes and ensuring seamless collaboration.

Historically, the execution of Python scripts within the department has been a decentralized and ad-hoc affair. Scripts are often stored on individual workstations or shared drives, making version control and access management cumbersome tasks. Furthermore, the reliance on Python as the scripting language presents challenges for non-technical users who may lack the necessary environment setup to execute these scripts.

The need for a centralized, user-friendly solution to script management and execution becomes evident in this context. By consolidating scripts into a GitHub repository and providing a web-based interface for their execution, the department aims to overcome these challenges and unlock new efficiencies in its workflow.

Moreover, the decision to leverage GitHub as the version control platform aligns with broader trends in collaborative software development and open science initiatives. GitHub's robust features for code management, issue tracking, and collaboration make it an ideal choice for fostering transparency and reproducibility within the department's research and development efforts.

In summary, the background of the project underscores the critical role of Python scripts in the radiation oncology workflow and the imperative to modernize and streamline their management and execution. By embracing web technologies and version control best practices, the department seeks to empower its team members to work more efficiently and collaboratively towards the common goal of advancing cancer care.

Goals and Motivation

Goals:

1. **Streamline Script Execution:** Develop a user-friendly web interface that enables easy execution of Python scripts without requiring Python installation. This interface should be intuitive for users with diverse technical backgrounds.
2. **Centralize Script Management:** Establish a GitHub repository to organize scripts, enabling version control, collaboration, and centralized access. This ensures that the latest versions of scripts are readily available and changes are tracked systematically.
3. **Enhance Scalability and Maintenance:** Design the dashboard and GitHub repository to accommodate future script additions and categorization. The system should be scalable and easy to maintain, allowing for seamless integration of new scripts and updates.
4. **Ensure Security and Compliance:** Implement security measures to protect sensitive patient data and ensure compliance with regulatory requirements. Access control mechanisms should be in place to restrict unauthorized access to scripts and data.

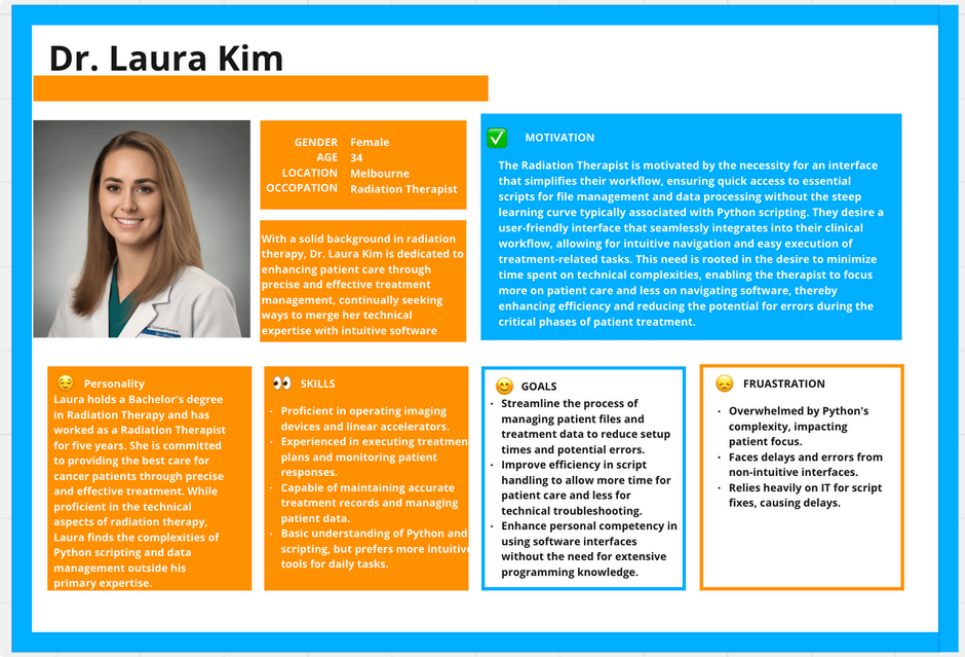
Motivation:

1. **Enhanced Efficiency:** The current decentralized approach to script management leads to inefficiencies in accessing, updating, and executing scripts. By centralizing script storage and providing a web-based interface for execution, the department aims to reduce the time and effort required to perform routine tasks, thereby increasing overall productivity.
2. **Improved Accessibility:** Many of the scripts used within the department require a Python environment for execution, posing a barrier for non-technical users who may lack the necessary software setup. By providing a user-friendly web interface for script execution, individuals across diverse skill sets can easily access and utilize these scripts without the need for Python installation.
3. **Version Control and Collaboration:** Without a centralized repository and version control system, tracking changes to scripts and collaborating on development becomes challenging. By leveraging GitHub, the project aims to establish a robust version control mechanism that enables collaboration, tracks changes, and ensures that team members are always working with the latest versions of scripts.
4. **Scalability and Maintenance:** As the department evolves and new scripts are developed, the need for a scalable and maintainable solution becomes increasingly apparent. By designing the web interface and GitHub repository to accommodate future additions and updates, the project lays the foundation for long-term scalability and ease of maintenance.
5. **Security and Compliance:** Handling sensitive patient data requires adherence to strict security and compliance standards. By centralizing script management within a secure GitHub repository, the department can implement access control measures to ensure that only authorized personnel have access to scripts and patient data, thereby mitigating the risk of data breaches or unauthorized use.
6. **Innovation and Collaboration:** By modernizing its script management infrastructure, the department can foster a culture of innovation and collaboration among team members. The centralized repository and web interface provide a platform for sharing ideas, collaborating on script development, and driving continuous improvement in cancer treatment processes.

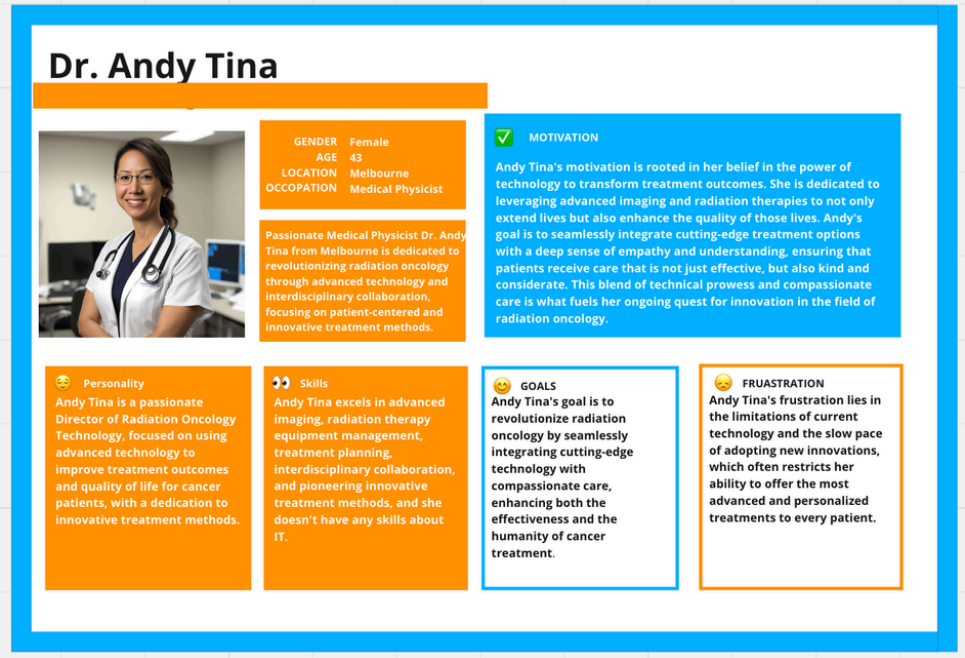
Personas

1. **Clinical Oncologist (Dr. Patric Wang)**
   - Background: Experienced oncologist specializing in radiation therapy.
   - Role: Oversees patient treatment plans and relies on accurate data management.
   - Needs: Access to scripts for data analysis and management, easy-to-use interface, assurance of data security and integrity.
   -

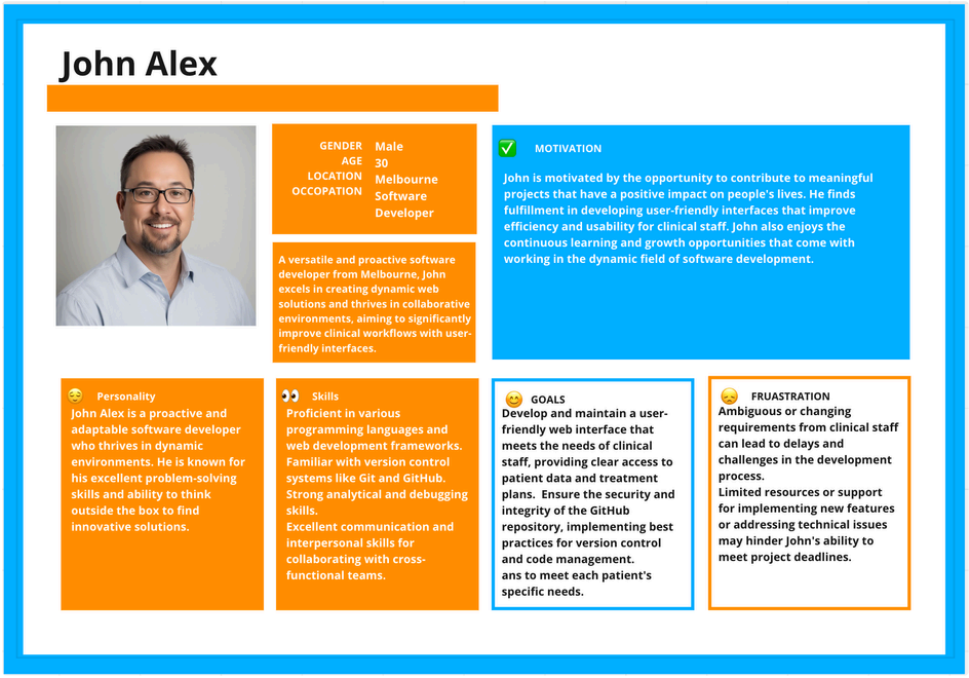
2. **Radiation Therapist (Dr. Laura Kim)**
   - Background: Clinical staff responsible for operating imaging devices and linear accelerators.
   - Role: Executes treatment plans and manages patient data during therapy sessions.
   - Needs: Quick access to scripts for file management and data processing, user-friendly interface compatible with clinical workflow.
   -



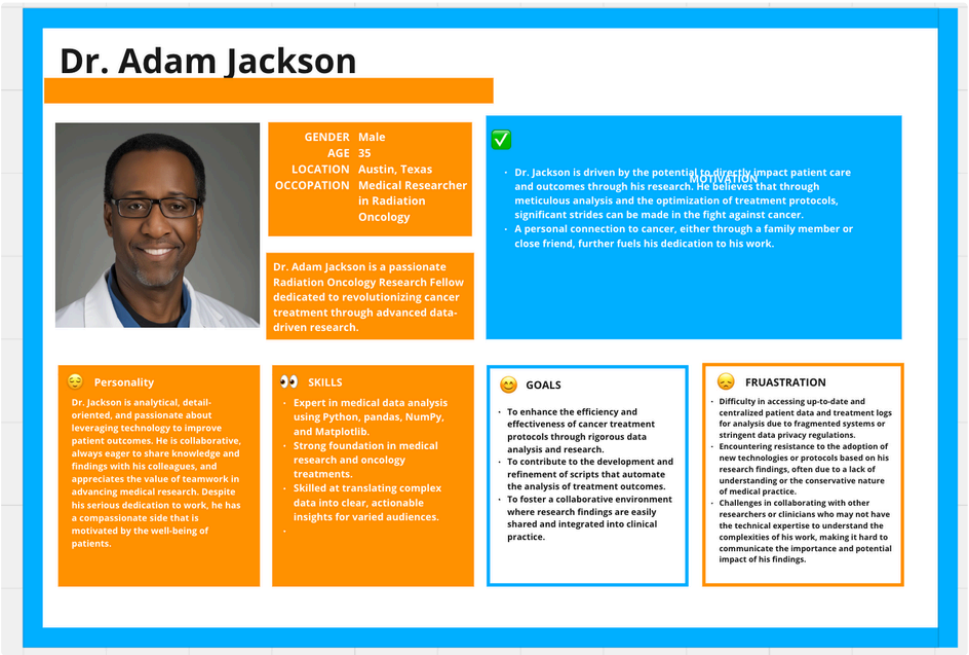
3. **Medical Physicist (Dr. Andy Tina)**
   - Background: Physics expert specializing in radiation therapy technology.
   - Role: Ensures the safe and accurate delivery of radiation doses to patients.
   - Needs: Version control for scripts, integration with existing hospital systems, ability to customize scripts for specific research or clinical needs.
   -



4. **Software Developer (John Alex)**
   - Background: IT professional with experience in software development.
   - Role: Develops and maintains the web interface and GitHub repository.
   - Needs: Clear requirements from clinical staff, flexibility to adapt to changing needs, collaboration tools for team communication and project management.
   -



5. **Research Fellow (Dr. Adam Jackson)**
   - Background: Medical researcher interested in analyzing treatment outcomes and optimizing protocols.
   - Role: Conducts research projects using patient data and treatment logs.
   - Needs: Access to scripts for data analysis and visualization, ability to contribute to script development and share findings with the team.



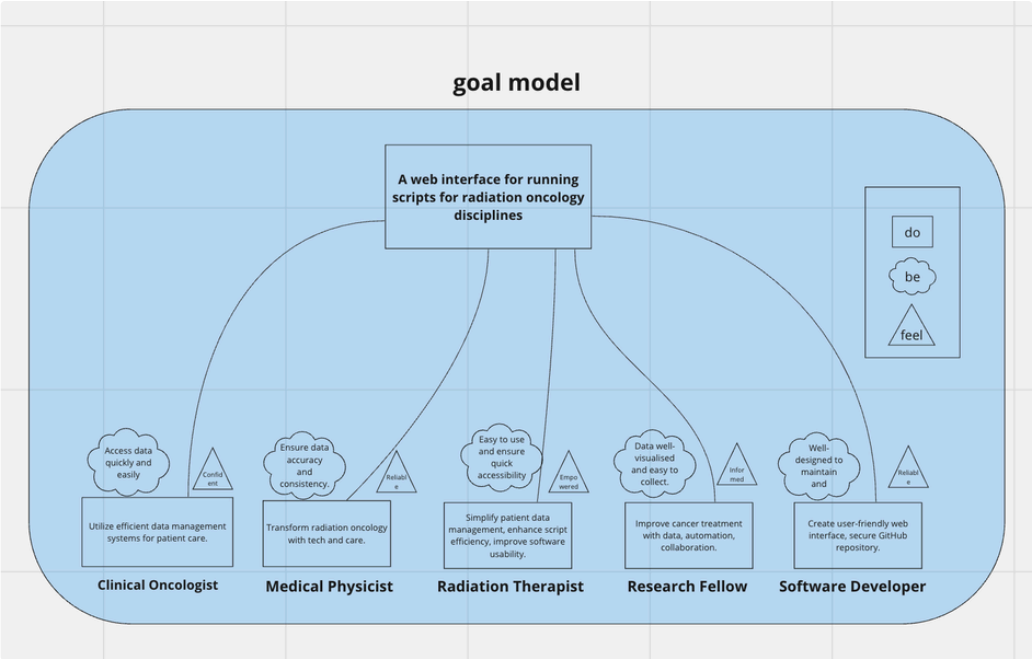
DO-BE-FEEL and GOAL MODEL

DO-BE-FEEL List:

do / be / feel

Roles	Do(Functional Goal)	Be(Quality Goal)	Feel(Emotional Goal)
Clinical Oncologist (Dr. Patric Wang)	Utilize efficient data management systems for patient care.	Access data quickly and easily.	Confident
Medical Physicist (Dr. Andy Tina)	Transform radiation oncology with tech and care.	Ensure data accuracy and consistency.	Reliable
Radiation Therapist (Dr. Laura Kim)	Simplify patient data management, enhance script efficiency, improve software usability.	Easy to use and ensure quick accessibility.	Empowered
Research Fellow (Dr. Adam Jackson)	Improve cancer treatment with data, automation, collaboration.	Data well-visualised and easy to collect.	Informed
Software Developer (John Alex)	Create user-friendly web interface, secure GitHub repository.	Well-designed to maintain and update.	Reliable

GOAL Model:


goal model
A web interface for running scripts for radiation oncology disciplines
do
be
feel
Access data quickly and easily
Confident
Utilize efficient data management systems for patient care.
Clinical Oncologist
Ensure data accuracy and consistency.
Reliable
Transform radiation oncology with tech and care.
Medical Physicist
Easy to use and ensure quick accessibility
Empowered
Simplify patient data management, enhance script efficiency, improve software usability.
Radiation Therapist
Data well-visualised and easy to collect.
Informed
Improve cancer treatment with data, automation, collaboration.
Research Fellow
Well-designed to maintain and
Reliable
Create user-friendly web interface, secure GitHub repository.
Software Developer

Project Architecture

In this section, project architecture is illustrated including

Technologies used in this project,

database schema,

website sucurity issue,

ethical consideration,

project UML

Technologies

For the development of our website, we have opted for a decoupled architecture, using TypeScript for the front-end and Python for the back-end. This approach allows us to leverage the strengths of both languages, ensuring a robust and scalable solution.

TypeScript has been chosen as the front-end language due to its ability to overcome several limitations of JavaScript. As a superset of JavaScript, TypeScript introduces strong typing, enhanced code structuring, and object-oriented features, which significantly reduce the likelihood of runtime errors and improve maintainability. This is particularly advantageous for projects regardless of their size, and it has gained substantial popularity within the Australian IT industry for its reliability and the robust tooling ecosystem.

On the back-end, we have implemented the Flask framework to meet the specific needs of our client. The client's familiarity with Flask, combined with its straightforward and lightweight nature, makes it an ideal choice for our project. Flask's ability to integrate seamlessly with other Python libraries and tools allows for quick development cycles and easy maintenance. This framework supports our needs for a flexible and efficient server-side solution that can easily adapt to changing requirements.

This technology stack not only facilitates a smooth development process but also ensures that both the client and our team can manage and scale the project effectively. The use of these modern, well-supported technologies positions us to handle the dynamic demands of web applications efficiently.

Frontend Technologies

Programming Language:

TypeScript

- **Role**: A superset of JavaScript providing optional static typing.
- **Benefits**: Facilitates building more robust software, while reducing the likelihood of common JavaScript errors at runtime.

Library:

1. **React and React-DOM**
    a. **Role**: Library for building user interfaces, particularly single-page applications where data updates frequently.
    b. **Benefits**: Facilitates the creation of reusable UI components, manages state effectively across components, and optimizes rendering performance.
2. **Next.js**
    a. **Role**: A React framework that provides server-side rendering, static site generation, and optimized bundling. It is used to build scalable and performant React applications.
    b. **Benefits**: Simplifies routing, API routes creation, and overall project structure. It also enhances performance with automatic code splitting and image optimization.

Styling and UI Frameworks

1. **Material-UI**
    - **Role**: A popular React UI framework with a comprehensive set of pre-designed components that follow Material Design principles.
    - **Benefits**: Accelerates development by providing ready-to-use components like buttons, cards, dialogs, etc., which are customizable and responsive.

Additional Libraries and Utilities

- **Axios**: Promise-based HTTP client for making requests to back-end services.
- **DayJS**: Lightweight date-time library, used for parsing, validating, and manipulating dates and times.

Backend Technologies

1. **Flask**
   - **Role**: Flask is a lightweight WSGI web application framework in Python, designed to be easy to use and extend, making it ideal for building simple web applications quickly and scaling up to complex applications.
   - **Benefits**: Since Flask is not prescriptive about database or ORM, it allows developers to choose their tools and libraries, which provides flexibility in building the application architecture.
2. **Flask-JWT-Extended**
   - **Role**: This extension provides JWT (JSON Web Tokens) support for Flask applications, facilitating secure and scalable authentication mechanisms.
   - **Benefits**: Offers features like token refreshing, token revocation, and additional security settings that are crucial for modern API security.
3. **SQLAlchemy**
   - **Role**: SQLAlchemy is an ORM (Object Relational Mapping) tool that allows Python applications to communicate with databases using high-level abstractions.
   - **Benefits**: Manages database schemas and performs database operations in a Pythonic way, reducing the need for raw SQL unless necessary and boosting productivity.

Additional Libraries and Utilities

- **Flask-Cors**: A Flask extension for handling Cross-Origin Resource Sharing (CORS), making it safe to enable cross-origin requests in the Flask app when necessary, especially for APIs consumed by different domains.

Development and Operations

- **Docker (Dockerfile and docker-compose.yml)**: These files suggest that Docker is used to containerize the application, ensuring it runs consistently across different environments.
- **Blinker**: Provides support for Signal handling in Python, which can be used to handle events in the application, like modifications in the database or certain actions triggering other actions.

Cloud Database: Supabase

Supabase is an open-source Firebase alternative that provides a suite of tools including a PostgreSQL database, authentication, instant APIs, real-time subscriptions, and storage. It's designed to simplify the backend development process, offering many Firebase-like features but with the flexibility of a traditional SQL database.

Database schema

Database relationtionship diagram is shown below.

Database url:

```
postgresql+psycopg2://postgres.bgniwlymuiboukuqbaha:LFofHUle88xRS6sv@aws-0-ap-southeast-2.pooler.supabase.com:543
```

```

-- existed and delete
DROP TABLE IF EXISTS user CASCADE;
DROP TABLE IF EXISTS script CASCADE;
DROP TABLE IF EXISTS execution CASCADE;
DROP TABLE IF EXISTS favorite CASCADE;


-- Create 'user' table
CREATE TABLE user (
    id SERIAL PRIMARY KEY,
    username VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    role_id INT,
    last_login TIMESTAMP,
```

```
    created_by VARCHAR(255),
    avatar BYTEA,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Create 'script' table
CREATE TABLE script (
    id SERIAL PRIMARY KEY,
    user_id INT,
    file_path TEXT NOT NULL,
    status VARCHAR(50),
    description TEXT,
    instruction TEXT,
    name VARCHAR(255),
    code TEXT,
    label TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    upload_required BOOL DEFAULT FALSE
);


-- Create 'execution' table
CREATE TABLE execution (
    id SERIAL PRIMARY KEY,
    script_id INT,
    user_id INT,
    start_time TIMESTAMP,
    end_time TIMESTAMP,
    output TEXT,
    status VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Create 'favorite' table
CREATE TABLE favorite (
    id SERIAL PRIMARY KEY,
    user_id INT,
    script_id INT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

```

Web Security

Record of security measures for the project：

Security Measure	Description	Implementation Date	Responsible Person	Review Date	Notes
Password Encryption	Use the encryption technology and methods to encrypt user passwords.	2024-04-19	@Zhihao Liang	2024-04-26	Regularly check encryption strength
Environment Variable for Database URL	Detailed explanation of how environment variables are used to store database URLs and other sensitive configurations to avoid hard-coding in the code.	2024-04-19	@Zhihao Liang	2024-04-26	Review the security of environment variables
Library Updates	Details the policy and process for regularly updating the libraries used in the project, including the tools used and the frequency of updates.	2024-04-19	@Lingyi Kong	2024-04-26	Focus on security update releases
JWT Authentication	Introduces the use of JWT for user authentication, including the processes for generating and verifying tokens.	2024-04-19	@Yuncong Ji	2024-04-26	Monitor vulnerabilities in authentication mechanisms

Details:

1. **Encrypting Passwords**: By encrypting passwords, you ensure that even if data is intercepted or accessed by unauthorised individuals, the passwords remain protected. This helps prevent unauthorised access and mitigates potential damage.
2. **Using Environment Variables for Database URLs**: Storing sensitive information such as database URLs in environment variables, rather than in your codebase, reduces the risk of exposing them in source control or to unauthorized access through other means. This is a standard practice for maintaining the security of database connections.
3. **Using Latest Libraries**: Keeping your libraries up-to-date is crucial because many updates include patches for security vulnerabilities that have been discovered since the last version. Regular updates help protect your systems from known attacks that exploit these vulnerabilities.

4. **Using JWT for Authentication**: JSON Web Tokens (JWT) are a compact, URL-safe means of representing claims to be transferred between two parties. Using JWTs can help in securely transmitting information between the client and server, which is especially useful for stateless authentication.

Maintenance:

1. **Regular Review**: The team should review the document on security measures at least once a week to ensure all information is up-to-date.
2. **Change Management**: Any changes or additions to security practices must be updated in this document. Changes must be reviewed and approved by the security team first.
3. **Access Control**: Only team members can edit this document; all other people will have read-only access.
4. **Feedback Mechanism**: Team members are encouraged to offer improvement suggestions. Feedback will be collected via team meetings or the internal email system.

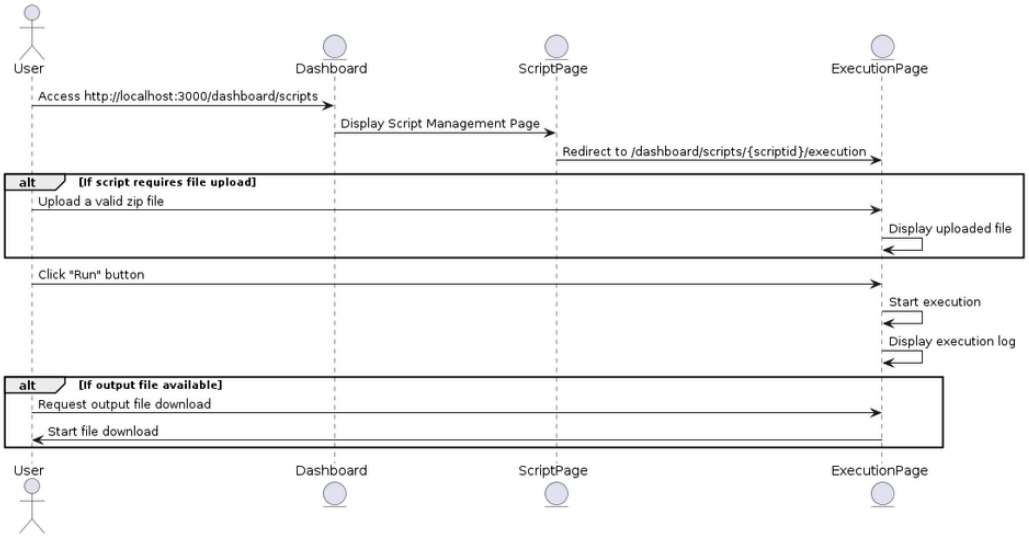
Ethical Considerations

Ethical Issues and Considered

1. **Privacy**:
   - **Concern**: User scripts may contain sensitive information, which could be inadvertently exposed or misused.
   - **Resolution**: Implement rigorous data handling protocols to ensure that all user data is encrypted and securely stored. Access to this data is restricted to authorized personnel only.
   - **Justification**: Protecting user data is a fundamental ethical obligation that also helps in building trust with our users.
2. **Transparency**:
   - **Concern**: Users need to understand how their data and scripts are being used and what the outputs imply.
   - **Resolution**: Provide clear documentation and user agreements that outline how data is handled, the purpose of data collection, and how outputs are generated.
   - **Justification**: Transparency is key to ethical user interactions and helps in ensuring that users are making informed decisions.
3. **Honesty**:
   - **Concern**: Accurate representation of what the script execution can and cannot do.
   - **Resolution**: Ensure that all user communications are honest and clear about the capabilities and limitations of our service.
   - **Justification**: Honesty prevents misinformation and promotes a trustworthy relationship with users.
4. **Inclusivity**:
   - **Concern**: Ensuring that the website is accessible to all potential users, regardless of their technical skills or disabilities.
   - **Justification**: Inclusivity enhances the usability of the website for a broader audience and is a key ethical commitment.
5. **Sustainability**:
   - **Concern**: Efficient use of server resources to avoid unnecessary computational waste and reduce operational costs.
   - **Resolution**: Optimize server performance through effective load balancing, caching, and on-demand resource allocation. Implement efficient coding practices to minimize processing time.
   - **Justification**: Efficient resource management reduces costs and improves service reliability and performance, aligning with ethical standards for responsible technology use.

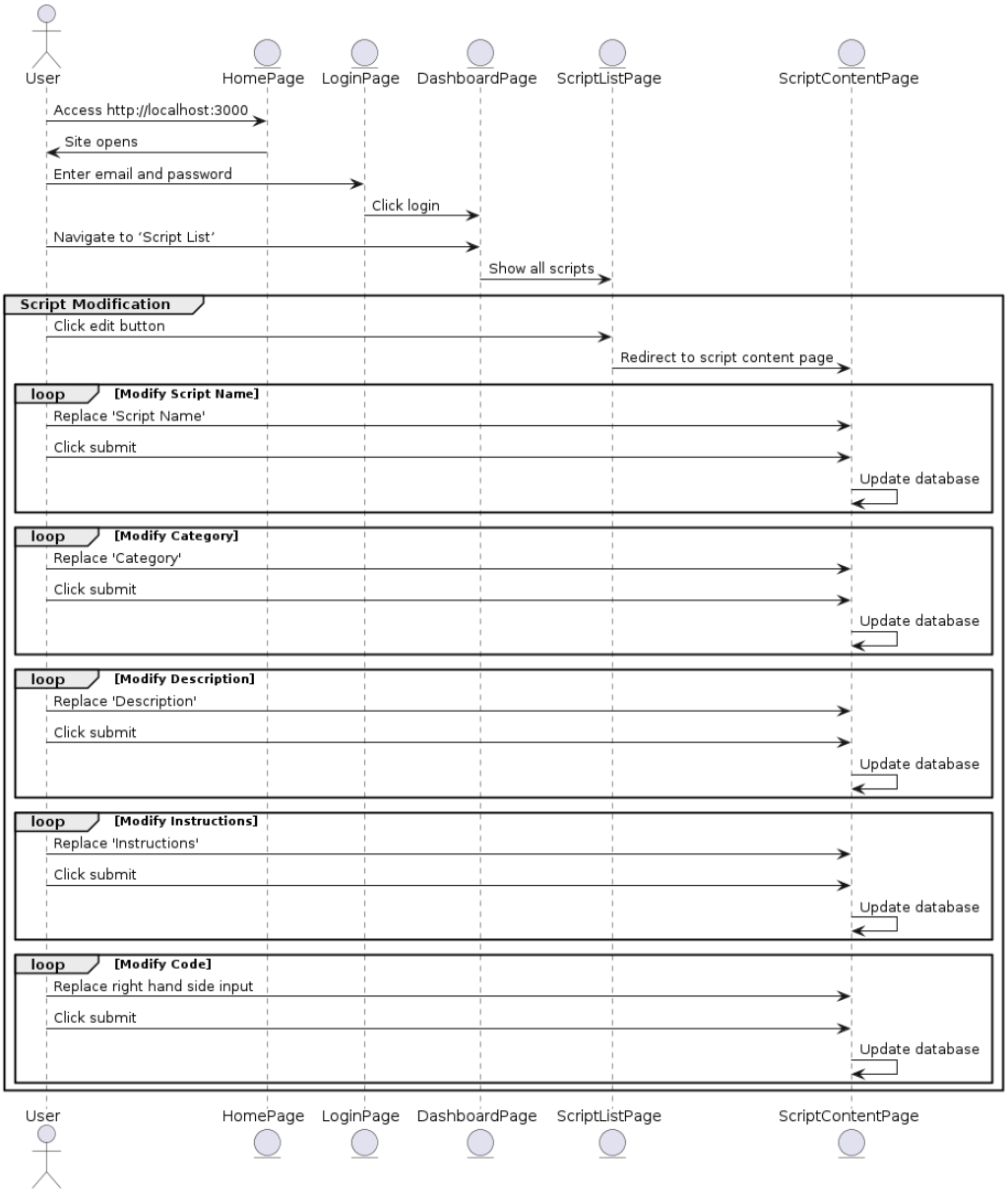
Project UML

Script Execution SD




```
@startuml
actor User
entity Dashboard
entity ScriptPage
entity ExecutionPage

User -> Dashboard: Access http://localhost:3000/dashboard/scripts
Dashboard -> ScriptPage: Display Script Management Page

ScriptPage -> ExecutionPage: Redirect to /dashboard/scripts/{scriptid}/execution

alt If script requires file upload
    User -> ExecutionPage: Upload a valid zip file
    ExecutionPage -> ExecutionPage: Display uploaded file
end

User -> ExecutionPage: Click "Run" button
ExecutionPage -> ExecutionPage: Start execution
ExecutionPage -> ExecutionPage: Display execution log

alt If output file available
    User -> ExecutionPage: Request output file download
    ExecutionPage -> User: Start file download
end

@enduml

```

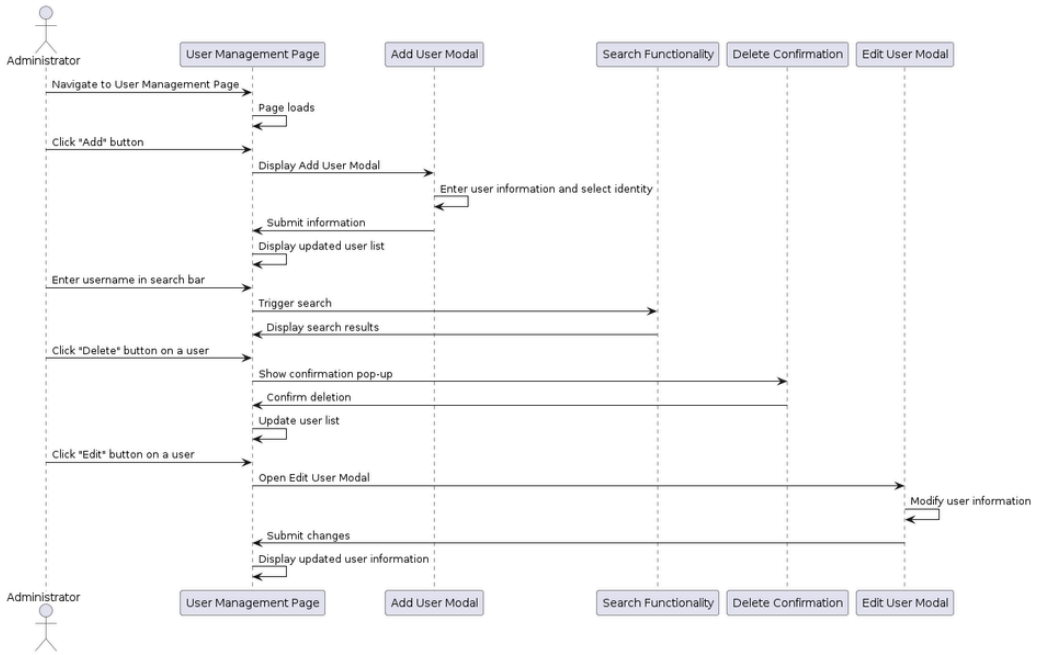
Edit Script SD




```
@startuml
actor User
entity HomePage
entity LoginPage
entity DashboardPage
entity ScriptListPage
entity ScriptContentPage

User -> HomePage: Access http://localhost:3000
HomePage -> User: Site opens

User -> LoginPage: Enter email and password
LoginPage -> DashboardPage: Click login
```

```
User -> DashboardPage: Navigate to ‘Script List’
DashboardPage -> ScriptListPage: Show all scripts

group Script Modification
    User -> ScriptListPage: Click edit button
    ScriptListPage -> ScriptContentPage: Redirect to script content page

    loop Modify Script Name
        User -> ScriptContentPage: Replace 'Script Name'
        User -> ScriptContentPage: Click submit
        ScriptContentPage -> ScriptContentPage: Update database
    end

    loop Modify Category
        User -> ScriptContentPage: Replace 'Category'
        User -> ScriptContentPage: Click submit
        ScriptContentPage -> ScriptContentPage: Update database
    end

    loop Modify Description
        User -> ScriptContentPage: Replace 'Description'
        User -> ScriptContentPage: Click submit
        ScriptContentPage -> ScriptContentPage: Update database
    end

    loop Modify Instructions
        User -> ScriptContentPage: Replace 'Instructions'
        User -> ScriptContentPage: Click submit
        ScriptContentPage -> ScriptContentPage: Update database
    end

    loop Modify Code
        User -> ScriptContentPage: Replace right hand side input
        User -> ScriptContentPage: Click submit
        ScriptContentPage -> ScriptContentPage: Update database
    end
end

@enduml

```

User management SD




```
@startuml
actor Administrator as admin
participant "User Management Page" as UMP
participant "Add User Modal" as AUM
participant "Search Functionality" as SF
participant "Delete Confirmation" as DC
participant "Edit User Modal" as EUM

admin -> UMP : Navigate to User Management Page
UMP -> UMP : Page loads

admin -> UMP : Click "Add" button
UMP -> AUM : Display Add User Modal
AUM -> AUM : Enter user information and select identity
AUM -> UMP : Submit information
UMP -> UMP : Display updated user list

admin -> UMP : Enter username in search bar
UMP -> SF : Trigger search
SF -> UMP : Display search results

admin -> UMP : Click "Delete" button on a user
UMP -> DC : Show confirmation pop-up
DC -> UMP : Confirm deletion
UMP -> UMP : Update user list

admin -> UMP : Click "Edit" button on a user
UMP -> EUM : Open Edit User Modal
EUM -> EUM : Modify user information
EUM -> UMP : Submit changes
```

```
UMP -> UMP : Display updated user information

@enduml

```

Requirements

User Stories

How to run scripts shared by the client

User Stories

Key content

- Summary
- User story classification
- Product backlog

According to our clients needs, product backlog is implement with epics: script execution and monitoring; script management and integration, and user access and security, and user experience and dashboard design. Product backlog is designed including prioity and size estimation, which can be considered as a requirement of project with acception criteria for each user story. The classification of prioity and size estimation is defined as following.

User stories classification:

Size Estimation	
Small	User stories that can be completed in less or equal than 1 day by one person.
Medium	User stories that can be completed in 2 to 4 days by one person.
Large	User stories that can be completed in 5 to 7 days by one person.

Priority	
Must have	Non-negotiable features that must be implemented.
Should have	Important features that can add significant value.
Could have	Features that can add value but impact a little if left out.
Will not have	Features that are not a need for the project in this specific time-frame.

Product Backlog

	Epic	As	I want to	So that	Priority	Size Estimation	Justification
1	Script Execution	Radiation therapist	Run a specific script from a web interface easily	I can perform analysis tasks using python from any location without local installations or configurations	Must have	Large	It's the initial phase of the entire project for both the frontend and backend. This stage involves time-consuming planning on how to implement script execution and ensure that long-running Python scripts can run without disconnecting.

2		Radiation therapist	Support multiple files or folders uploading	I can analysis multiple parents information simultaously	Must have	Large	Script execution, highly demanded by clients, requires the capability to upload multiple files. We will need to design a way to modify file paths, taking into account that scripts may specify particular path names.
3		Radiation therapist	Obtain the output of scripts	I can get wanted one or lots of processed pateints information	Must have	Small	Users require modifications to patient information. It is straightforward to implement generated file downloads within the existing framework.
4		IT departmeнt developer	Website to integrate smoothly with SQL-based databases	I can manage data consistently with other applications we use.	Must have	Small	The IT department of the client company has requested an easy-to-use SQL-based database, a common requirement in the technology sector.
5	Script Management	Radiation therapist	Have a user-friendly script execution dashboard	I can easily find and run the script I want	Must have	Large	It's important to provide clients with a list of scripts so they can easily operate the system.
6		Radiation therapist	Access detailed usage instructions for each script directly	I can use scripts efficiently and correctly without extensive training	Must have	Small	It's essential to provide instructions to users, particularly for those who have no programming knowledge.
7		Radiation therapist	Be able to save frequently used scripts	I can easily and conveniently execute the scripts I need	Must have	Medium	If the number of scripts increases, it becomes necessary to implement a save function for specific scripts. This will involve UI design and database integration.
8		Radian Researchers	version control all my scripts	I access history scripts and manage all scripts easily	Must have	Small	As the client specifically points out needs, it must be done. Github repo can easily achive this needs
9	User access	Department administrator	Have two groups of users, administrator and normal user	Coworkers with proficient coding knowledge can upload, edit and delete needed scripts and the other group can focus on script execution	Must have	Large	As the client has specifically outlined their needs, these must be addressed. A GitHub repository can easily fulfill these requirements.
10		Department administrator	Let different users access the corresponding system based on their role	They can access different features based on their permission.	Must have	Medium	Implementing role-based access control is essential for maintaining system security and ensuring users have the appropriate permissions to perform their tasks efficiently and securely. This setup prevents unauthorized access and optimizes

							the user experience by providing role-specific functionalities.
11		Department administrator	Have a role of administrator for user account management	The efficient management of users can be handles within teams.	Must have	Large	We need to design the UI and implement backend changes for improved user management, as specifically requested by the client.
12	Script Management	Medical physicist	Have a dashboard that displays the status of scripts	I can have orginased space to run script and see which script to be run.	Should have	Medium	The client may not run scripts simultaneously. Additionally, UI design modifications and updates to the current framework are required.
13		Radiation researcher	Categorize and label additional scripts for future integration	The system can easily scale up to 20 scripts without confusing the users	Could have	Medium	It's easy for users to search for the script they want. Need Ui design and backend search functionality support for quick search
14		Medical physicist	Customize my dashboard to highlight frequently used scripts and critical information	I can streamline my daily workflow and improve efficiency.	Could have	Large	Modularizing the UI could enhance the user experience, but it requires careful design and may significantly alter the previous layout.
15		IT Department developer	Have a comprehensive deployment guide for the Flask-based web interface	I can efficiently set it up on our servers without having to learn new technologies.	Could have	Small	Considering the website needs to be deployed on the client's server, detailed documentation should be written.
16		Clinical Oncologist	Schedule scripts to run at specific times or intervals	Routine tasks can be automated, reducing manual effort and increasing consistency.	Could have	Medium	Ranked as the 'could have' feature of medium priority is scheduling scripts to run at certain times or intervals as it drastically automates repetitive activities and increases operational efficiency, however, it is not a core functionality and may bring much value in daily workflow optimization

How to run scripts shared by the client

Download the latest version data shared by the client from Google Drive

Script 1 Copy Monaco Patient

1. Replace the original code with below code

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 18 08:33:17 2022

@author: MCDELZ
"""

import os, re
import shutil
import ctypes  # An included library with Python install.
from sys import exit


current_folder_path = os.getcwd()
# this is scripted, better to ask user to select 1 or multiple target plan folders & destinations to copy & cha
location1 = f"{current_folder_path}{os.sep}location1"
location2 = f"{current_folder_path}{os.sep}location2"

# list of plan folders to change, format: P[[old name1, new name1], [old name2, new name2]]
P = [ ['zzACDSU','RzzACDSUtest'] ]


# first copy folder from location 1 to location 2, then edit location 2

for pat in P:    # pat = P[0]

    IDold = pat[0]
    IDnew = pat[1]


     # 1 ---- folder name----
    path1 = location1 + os.sep + '1~' + IDold + os.sep
    path2 = location2 + os.sep + '1~' + IDnew + os.sep

    if os.path.exists(path2):
        # ctypes.windll.user32.MessageBoxW(0, "The new plan folder already exists", "Not copied", 0)
        print("The new plan folder already exists")
        exit()
    else:
        shutil.copytree(path1, path2)




    # 2 ---- demog.xxx
    path = location2 + os.sep + '1~' + IDnew + os.sep

```

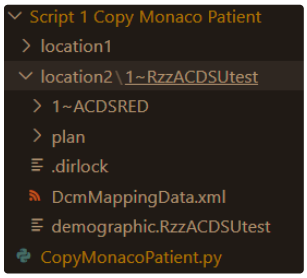
```
    old = path + 'demographic.'+ IDold
    new = path + 'demographic.'+ IDnew
    if os.path.exists(old):
        os.rename(old, new)

        with open(new, 'r') as filepointer:
            filedata = filepointer.read() #open plan file

        filedata2  = filedata.replace('\n' + IDold + '\n', '\n'+IDnew+'\n') #replace ID

        with open(new, 'w') as filepointer:
            filepointer.write(filedata2) #write new plan file



    # 3 ---- plan / xxx.hyp----

    path = location2 + os.sep + '1~' + IDnew + os.sep + 'plan'

    plist = os.listdir(path)

    for planfolder in plist:
        old = path + os.sep + planfolder + os.sep + IDold + '.hyp'

        if os.path.exists(old):
            new = path + os.sep + planfolder + os.sep + IDnew + '.hyp'
            os.rename(old, new)


        else:
            print(f"skip {old}")

        oldb = path + os.sep + planfolder + os.sep + IDold + '_rxB.hyp'
        if os.path.exists(oldb):
            newb = path2 + os.sep + planfolder + os.sep + IDnew + '_rxB.hyp'
            os.rename(oldb, newb)
        # else:

            # print(f"skip {oldb}")




    # 4 ---- plan/plan.txt > xxx----

    path = location2 + os.sep+'1~' + IDnew + os.sep + 'plan'

    plist = os.listdir(path)

    for planfolder in plist:
        planpath =   path + os.sep + planfolder + os.sep + 'plan'

        if os.path.exists(planpath):
            with open(planpath, 'r') as filepointer: filedata_old = filepointer.read()
            if filedata_old.find('\n'+IDold+'\n')>0:
                filedata_new  = filedata_old.replace('\n'+IDold+'\n', '\n'+IDnew+'\n') #replace ID
                with open(planpath, 'w') as filepointer: filepointer.write(filedata_new)
```

```
        planpathB =   path + os.sep + planfolder + os.sep + 'rxB_plan'
        if os.path.exists(planpathB):
            with open(planpathB, 'r') as filepointer: filedata_old = filepointer.read()
            if filedata_old.find('\n'+IDold+'\n')>0:
                 filedata_new  = filedata_old.replace('\n'+IDold+'\n', '\n'+IDnew+'\n') # replace ID
                 with open(planpathB, 'w') as filepointer: filepointer.write(filedata_new)









    # 5 ---- CT / info.txt > xxx----

    path = location2 + os.sep + '1~' + IDnew
    if os.path.exists(path):
        clist = [item for item in os.listdir(path) if re.match(r'[0-9]+~*', item)]

        for ct in clist:
            ctpath =   path + os.sep + ct + os.sep + 'info'

            with open(ctpath, 'r') as filepointer:
                filedata_old = filepointer.read()

            filedata_new  = filedata_old.replace('~'+IDold+'\n', '~'+IDnew+'\n') #replace ID
            filedata_new  = filedata_new.replace('\n'+IDold+'\n', '\n'+IDnew+'\n')

            with open(ctpath, 'w') as filepointer:
                filepointer.write(filedata_new)


    print(f"The new plan folder copied & changed!\n from: \n{path1}\n\nto: \n{path2}")
    # ctypes.windll.user32.MessageBoxW(0, f"The new plan folder copied & changed!\n from: \n{path1}\n\nto: \n{p
```

2. Run the script

```
python CopyMonacoPatient.py
```

3. Inspect the result in “location2” folder



Script 2 Reorganise Mim Export Data

Replace the code

```
# -*- coding: utf-8 -*-
"""
```

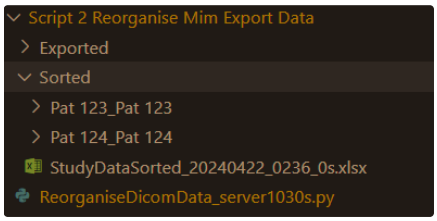
```
Created on Tue May 18 14:16:05 2021

@author: Leah & Jess
"""

# organise single folder level into ID - date_Scantype - scan name / RTPlan-Struct-Dose / other
# ignore PR, common text at start of scan name

# read main folder, for each folder get ID, name, date, scan type, scan name

# to do:
# keep 3 layers: L1 = patient, L2 = study description, L3 = series description / other
# read in export folder (month_Studies), copy to sorted folder, if patient exists, write scans & studies inside
# instead of index, create parameter table, write to excel

import os, time
import shutil
import pandas as pd
from   datetime import datetime


t1 = time.perf_counter()
def clean_path(path):
    path = path.replace('/',os.sep).replace('\\',os.sep)
    if os.sep == '\\' and '\\\\?\\' not in path:
        # fix for Windows 260 char limit
        relative_levels = len([directory for directory in path.split(os.sep) if directory == '..'])
        cwd = [directory for directory in os.getcwd().split(os.sep)] if ':' not in path else []
        path = '\\\\?\\' + os.sep.join(cwd[:len(cwd)-relative_levels]\
                        + [directory for directory in path.split(os.sep) if directory!=''][relative_levels:])
    return path

# sourcepath = r"H:\Restrict\Radiotherapy\DicomData\exported"
# sortedpath = r"H:\Restrict\\Testsort\testsortNonUnity"

# ***change to open dialogue, defaults to this one, but possible to change
sourcepath = r"\\server1030s\DataRegistry\MIM EXPORT\EXPORT Research_MR"
sortedpath = r"\\server1030s\DataRegistry\StudyDataSorted"
redcappath = r"\\server1030s\DataRegistry\StudyDataSorted\RedCap_MRRegistry_DATA_2023-04-14_0850.csv"

# root = tk.Tk()
# root.withdraw()
# sourcepath = filedialog.askopenfilename(initialdir=sourcepath)
current_folder_path = os.getcwd()
sourcepath = f"{current_folder_path}\Exported"

print('sourcepath:', sourcepath)


sortedpath = f"{current_folder_path}\Sorted"
IndexFile = sortedpath +os.sep+'StudyDataSorted'+ datetime.now().strftime("_%Y%m%d_%H%M")+".xlsx"
print('IndexFile:', IndexFile)
#*** read in RedCap file, use Mosaiq to get list of UR>Subfile, get list of IDs

os.chdir(sortedpath)

print(f'started {t1}')

```

```
sites = {   'abdo':         ['abdo', 'abdomen'],
            'abdo.upp':     ['abdomen upper','upper abdo',  'duod',  'stomach'],
            'abdo.lwr':     ['abdomen lower','lower abdo','colon'],
            'liver':        ['liver'],
            'kidney':       ['kidney'],
            'pancreas':     ['pancreas'],
            'brain':        ['brain'],
            'breast':       ['breast'],
            'chest':        ['chest'],
            'hn':           ['head','neck','HandN','HNbilat','HNipsi', 'H&N', 'H.&.N'],
            'lung':         ['lung'],
            'oesophagus':   ['oesoph','esoph'],
            'pelvis':       ['pelvis', 'pelvis general','bowel'],
            'rectum':       ['rectum'],
            'bladder':      ['bladder'], # bladder, bowel, uterus and cervix, vagina, and rectum
            'gynae':        ['gynae'],
            'prostate':     ['prostate', 'prost', 'pros'],
            'spine':        ['spine'],
            'pelvis.spine': ['pelvis&spine'],
            'bone':         ['femur']
            }

techs = {'stereo': ['stereo', 'sbrt', 'srs'],
        'pall': ['pall', 'palliative'],
        'FB.BH':['free', 'bh'],}


studymonths = os.listdir(sourcepath)
print(studymonths)

for studymonth in studymonths:
    #studymonth = studymonths[0] #test


    folders = os.listdir(sourcepath + os.sep + studymonth)
    # f = folders[2] # testing

    for f in folders:  # 0 = patname, 1 = patid, # 2 = modality, 3 = studydate, 4 = studytime 5 = studydesc, si
        # f = folders[0] #test
        HDx = {'patname':'', 'patid':'', 'modality':'', 'studydate':'', 'studytime':'',  'studydesc':'', 'serie
        for hd in list(HDx.keys()):
            HDx[hd] = f.split('_')[list(HDx.keys()).index(hd)]

        # *** check if ID is in the RedCap list, skip if not.




        usesite = 'sx'
        for sitegroup in list(sites.keys()):
            for site in sites[sitegroup]:
                if site in HDx['studydesc'].lower(): #'prostate': ['prostate', 'prost', 'pros','Prostate'],
                    usesite = sitegroup
        HDx['site'] = usesite

        usetech = 'tx'
        for techgroup in list(techs.keys()):
            for tech in techs[techgroup]:
```

```
                if tech in HDx['studydesc'].lower():
                    usetech = techgroup
        HDx['tech'] = usetech

        inst = HDx['instance']
        try: HDx['instance'] = int(inst)
        except: HDx['instance'] = inst.lstrip('0')



        #level thing:
        L1 = HDx['patid'] + '_' + HDx['patname']
        L2 = HDx['studydate'] + '_' + HDx['modality']  + '_' + HDx['site']   + '_' + HDx['tech']
        L3 = HDx['studytime'] + '_' + HDx['studydesc'] + '_' + HDx['series'] + '_' + HDx['nslices'] + '_' + str

        n = int( HDx['nslices'].split('n')[1]) # number of dicom files/slices
 #       if n<3 or len(HDx['series'])<2 or HDx['modality'] not in ['MR', 'CT', 'PR']:  L2 = 'other_'+L2 #JL cha
        if HDx['modality'] not in ['MR', 'CT', 'RTDOSE', 'RTst', 'RTPLAN', 'PT']:  L2 = 'other_'+L2

        scanfolder1 = sourcepath+ os.sep+studymonth+os.sep+f
        # print('scanfolder1:', scanfolder1)
        scanfolder2 =  sortedpath + os.sep + L1 + os.sep + L2 + os.sep + L3
        # print('scanfolder2:', scanfolder2)
        pat = HDx['patid'] + '-' + HDx['patname']  + ' '+  HDx['nslices']
        print(f'folder:{studymonths.index(studymonth)}/{len(studymonths)} study:{folders.index(f)}/{len(folders
        print(' ')

        if not os.path.exists(scanfolder2):   shutil.copytree( clean_path(scanfolder1) ,clean_path(scanfolder2)
        # else: print('skip')


t2 = time.perf_counter()
print(f"{(t2-t1)/60:0.1f}min for sorting" )

#%%

# ----------------------------------------
# create and excel file with index table
# ----------------------------------------
t3 = time.perf_counter()
HDx = ['PatientID', 'Name', 'other', 'StudyDate', 'Modality', 'Site', 'Technique',
       'StudyTime', 'StudyDescription','SeriesDescription','Nslices','Instance']

DicomList = pd.DataFrame(columns = HDx)
Flist = []

for root, rdirs, rfiles in os.walk(sortedpath):  # takes a while\

    level = root.replace(sortedpath, '').count(os.sep)

    if level == 3:
         # print(root)
         folder = root.replace(sortedpath, '').replace(os.sep, '_').split('_')
         # eg "\\server1030s\DataRegistry\StudyDataSorted\211230_RYAN^ANTHONY^FRANCIS\2021-11-18_MR_oesophagus_



         folder = folder[1:] #drop 1st empty value
         Flist.append(folder)
```

```
        ind = Flist.index(folder)
        if folder[2]!='other': folder.insert(2, '-')

        for i in range(len(HDx)):
            try: DicomList.loc[ind, HDx[i]] = folder[i]
            except: DicomList.loc[ind, HDx[i]] = '-'

#*** add Redcap data columns for each ID

t4 = time.perf_counter()

#%%

diff = t2-t1
txt = 'sort finished'
if diff<60:
    diffs = f'_{diff:0.0f}s'
    print(f"{txt}, took {diff:0.1f} seconds" )
elif diff<3600:
    diffs = f'_{diff/60:0.0f}min'
    print(f"{txt}, took {diff/60:0.1f} minutes" )
else:
    diffs = f'_{diff/3600:0.0f}h'
    print(f"{txt}, took {diff/3600:0.1f} hours" )
IndexFile = IndexFile.replace(".xlsx", f"{diffs}.xlsx")
DicomList.to_excel(IndexFile, sheet_name = 'DicomList')


diff = t4-t3
txt = 'table finished'
if diff<60:        print(f"{txt}, took {diff:0.1f} seconds" )
elif diff<3600:    print(f"{txt}, took {diff/60:0.1f} minutes" )
else:              print(f"{txt}, took {diff/3600:0.1f} hours" )
```

2. Run the script
3. Result: see “Sorted” folder



Script 3 Edit Monaco Template

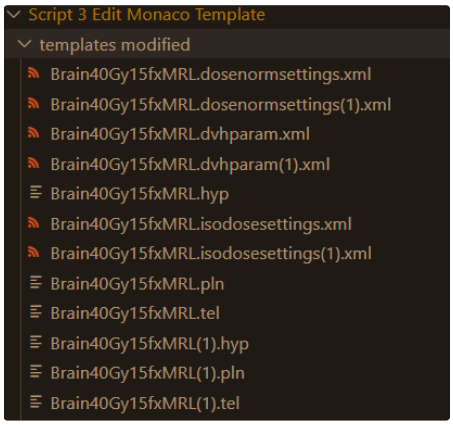
1. Replace the code

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 21 14:58:09 2022

@author: MCDELZ
"""

```

```
8
9
10  import os, re, time
11  from pathlib import *
12  import shutil
13
14
15  #list 00 files
16  # folderpath = "\\\\cmsada2\\FocalData\\MonacoTemplates"
17  # files00 = [f.split('.tel')[0] for f in os.listdir(folderpath) if f.endswith('.tel')]
18  # files00 = [f for f in files00 if f.startswith('00')]
19  t1 = time.perf_counter()
20
21  current_folder_path = os.getcwd()
22  editfolder = f"{current_folder_path}{os.sep}templates original"
23  donefolder = f"{current_folder_path}{os.sep}templates modified"
24
25  # rename
26  files = os.listdir(editfolder)
27
28  files = [f for f in files if os.path.isfile(editfolder + os.sep + f)] # remove folders
29
30  for f in files:
31      f2 = re.sub('New00', '00', f)
32
33      ind = f2.find('v0')
34      if ind>=0:
35          vsn = str(int(f2.split('v0')[1][0])+1)
36          f3 = re.sub(r'v0\d',  r'v0'+vsn, f2)
37      else:
38          f3 = f2
39      shutil.copy(editfolder + os.sep + f, donefolder + os.sep + f3) # write to done folder
40
41
42  t2 = time.perf_counter()
43  s = 'remove new, add 1 to v'
44  print(s + " (" + f"{t2-t1:0.1f}s)" )
45  t1 = time.perf_counter()
46
47
48
49
50  # remove :
51  # source = clinicfolder
52  source = donefolder
53  files = os.listdir(source)
54  files = [f for f in files if os.path.isfile(source + os.sep + f)] # remove folders
55  files = [f for f in os.listdir(source) if f.endswith('.tel')]
56
57  colonset = []
58
59  for f in files:
60      if f.endswith('.tel'):
61
62          telpath = source + os.sep + f
63
64          with open(telpath, 'r') as filepointer:   filedata = filepointer.read() #open plan file
65
```

```
        indset = [i for i in range(len(filedata)) if filedata.startswith(':',i)]

        colonset.append([f, indset])

        if len(indset)==2:
            for i in range(2):
                ind1 = indset[i] - filedata[indset[i]-3:indset[i]].find('\n')
                ind2 = indset[i] + filedata[indset[i]:indset[i]+50].find('\n')

                line1 = filedata[ind1:ind2]

                print(f"{f} \n{line1}") # show lines with :

            #edit & overwrite tel file

            filedata2  = filedata.replace(': ', ' ') #replace :
            filedata2  = filedata.replace(':', ' ') #replace :

            with open(telpath, 'w') as filepointer:  filepointer.write(filedata2) #write new plan file

t2 = time.perf_counter()
s = 'remove : from tel'
print( s + " (" + f"{t2-t1:0.1f}s)" )
t1 = time.perf_counter()
```

2. Run the script
3. Inspect the result
4. (templates modified)



Sprints

Sprint 1

- Sprint 1 Planning

Sprint 2

- Sprint 2 planning
- Sprint 2 Review
- Sprint 2 Retrospective

Sprint 1

Sprint 1 Planning

Sprint 1 Planning

Key content

- Sprint 2 plan
- Sprint 3 plan
- Technologies proposed to use
- Risk assessment
- Infrastructure to deploy

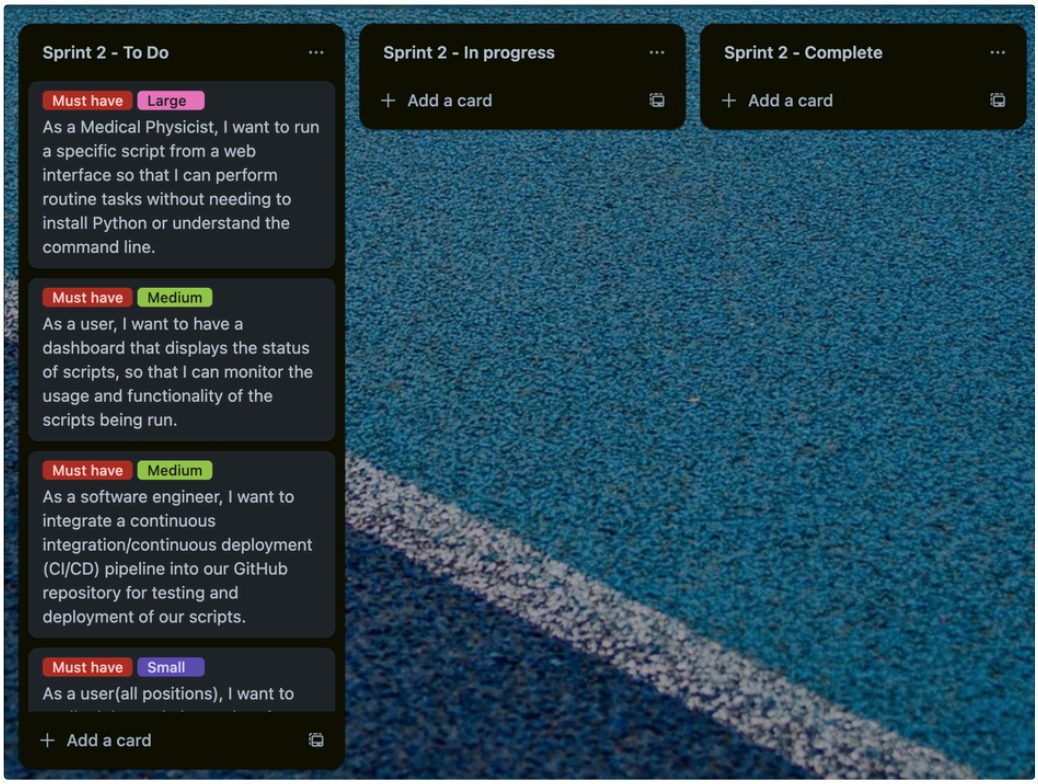
Link of our Trello board: FL-agile trello planning

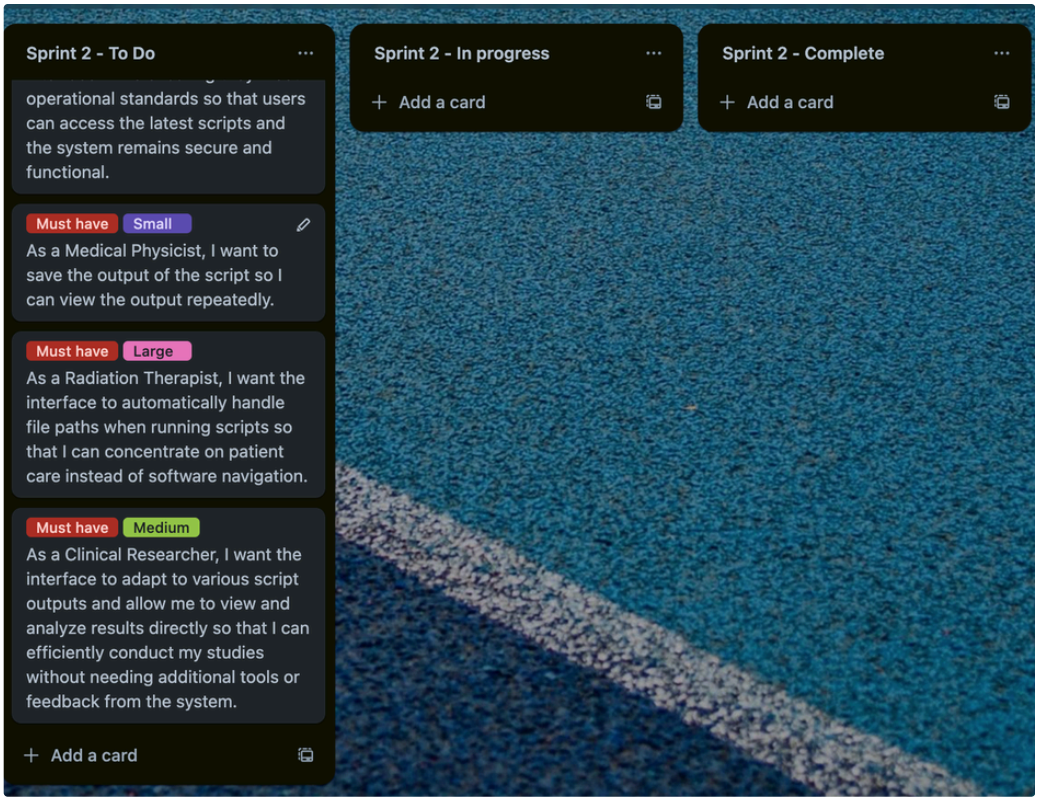
Sprint 2 Plan:

In Sprint 2, our focus is on implementing functionalities critical to the client's needs, primarily addressing user stories labeled as "Must have." with emphasizing epic of script execution and onitoring, script management and intergrate. Additionally, we'll discuss the foundational logic design for the wedsite, particularly the logic behind implementing file path changes for scripts, which is essential for our solution's architecture.

Trello Agile Sprint Board - FL

Development Requirements:





Technologies Used:

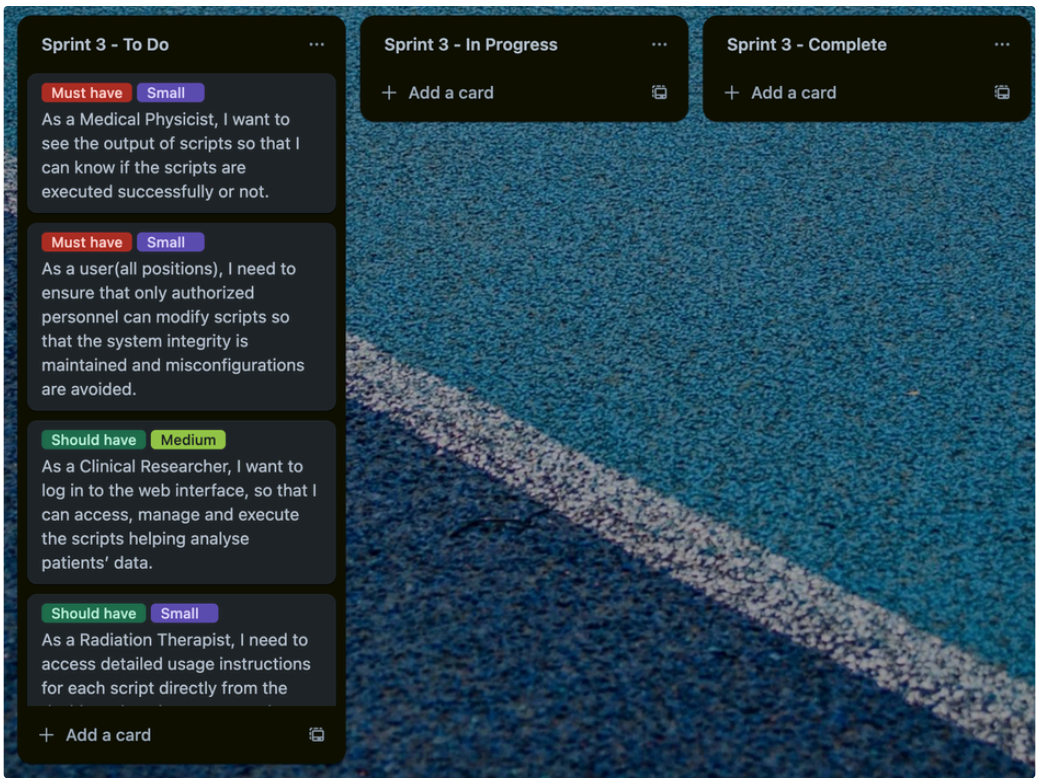
- Frontend: HTML, CSS, JavaScript (React framework)
- Backend: Flask (Python)
- Database: Mysql (for storing user information and script execution records)
- Deployment platform：Client's Healthcare Platform

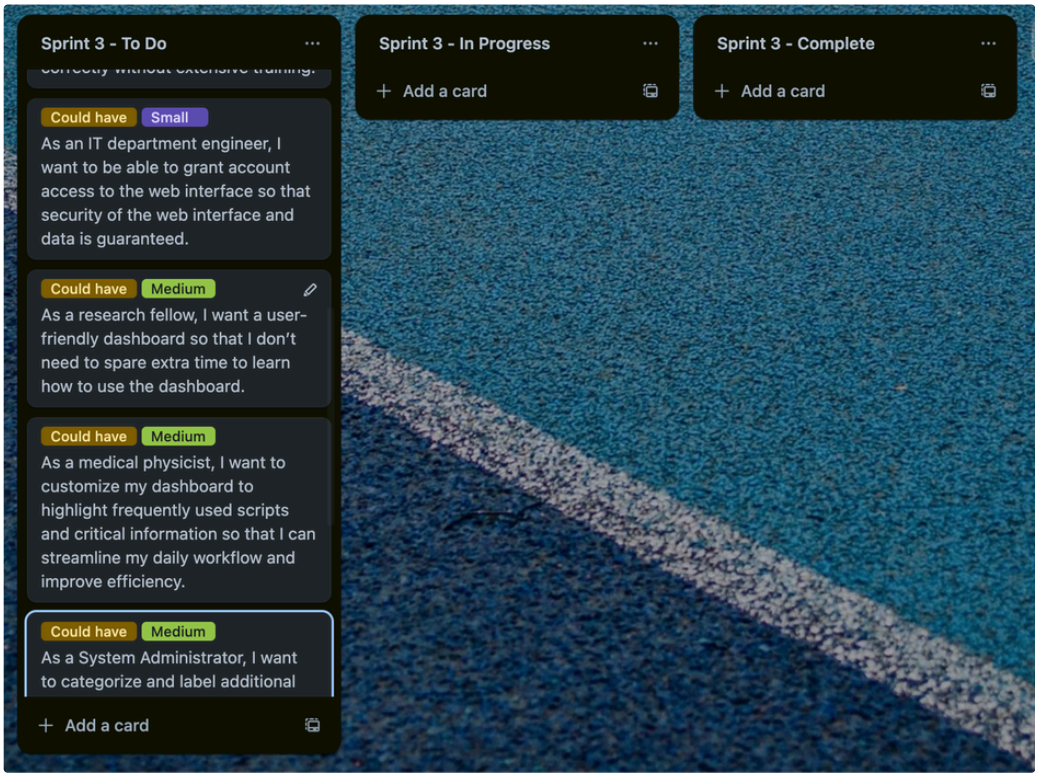
Sprint 3 Plan:

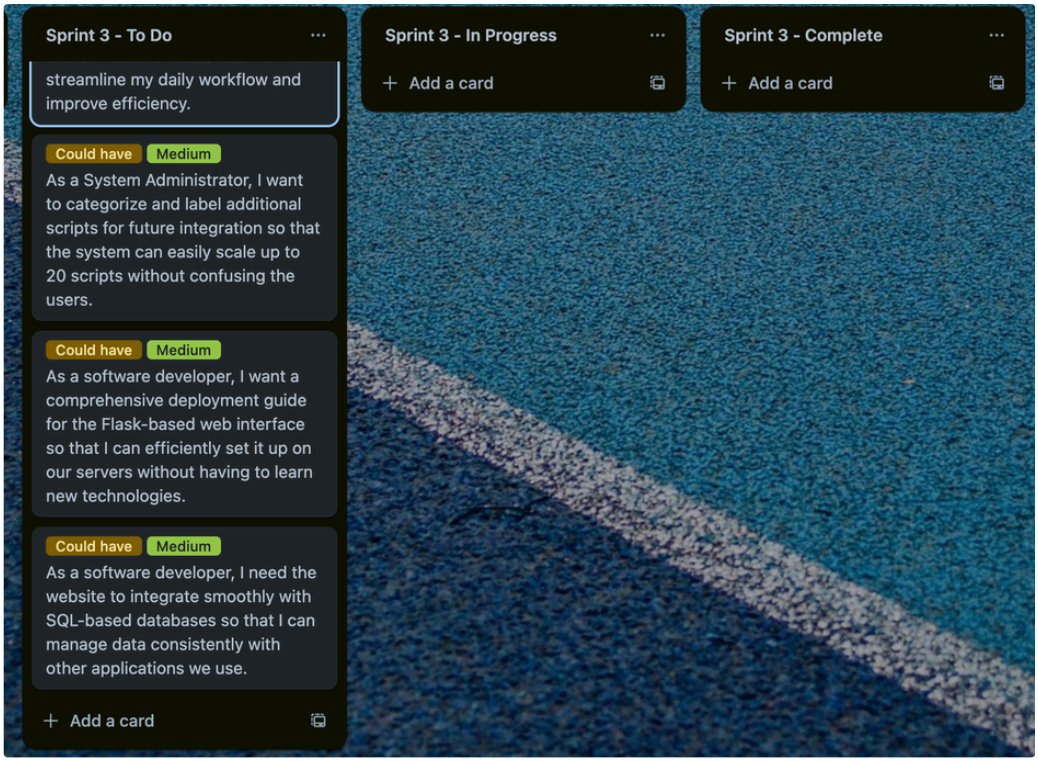
In Sprint 3, the primary focus will be on addressing the "Should Have" and "Could Have" tasks, specifically targeting improvements in the user experience and design of the dashboard. This sprint aims to enhance the overall usability and functionality of the dashboard interface, making it more intuitive and efficient for users. The team will concentrate on refining features, optimizing layout, and incorporating feedback to ensure the dashboard meets user expectations and facilitates smoother interaction with the system. By prioritizing these enhancements, we seek to elevate the user journey and provide a more engaging and effective platform for managing and monitoring tasks within the radiation oncology department's script execution and management system.

Development Requirements:

Trello Agile Sprint Board - FL


Sprint 3 - To Do
Must have Small
As a Medical Physicist, I want to see the output of scripts so that I can know if the scripts are executed successfully or not.
Must have Small
As a user(all positions), I need to ensure that only authorized personnel can modify scripts so that the system integrity is maintained and misconfigurations are avoided.
Should have Medium
As a Clinical Researcher, I want to log in to the web interface, so that I can access, manage and execute the scripts helping analyse patients' data.
Should have Small
As a Radiation Therapist, I need to access detailed usage instructions for each script directly from the
Add a card
Sprint 3 - In Progress
Add a card
Sprint 3 - Complete
Add a card


Sprint 3 - To Do
Could have Small
As an IT department engineer, I want to be able to grant account access to the web interface so that security of the web interface and data is guaranteed.
Could have Medium
As a research fellow, I want a user-friendly dashboard so that I don't need to spare extra time to learn how to use the dashboard.
Could have Medium
As a medical physicist, I want to customize my dashboard to highlight frequently used scripts and critical information so that I can streamline my daily workflow and improve efficiency.
Could have Medium
As a System Administrator, I want to categorize and label additional
Add a card
Sprint 3 - In Progress
Add a card
Sprint 3 - Complete
Add a card



Technologies Used:

- Frontend: HTML, CSS, JavaScript (React framework)
- Backend: Flask (Python)
- Database: Mysql (for storing user information and script execution records)
- Deployment platform：Client's Healthcare Platform

Risk Assessment

Given our situation, all team members are part-time developers, balancing internships or other coursework. This presents a risk: coordinating daily stand-up meetings may be challenging due to varying schedules, potentially impacting communication efficiency.

Furthermore, significant progress is anticipated mainly during weekends, which could delay client feedback and subsequently, our responses. This staggered working pattern necessitates a flexible approach to project management and communication.

Moreover, some team members are not well-versed in the Flask framework, as specified by the client, indicating a learning curve that could affect development speed and quality. This inexperience is a risk that we need to mitigate through additional training or allocating more time for those tasks.

Overall, we must adapt our development process to accommodate part-time availability and varying expertise levels, ensuring continuous progress and effective communication despite these challenges.

Infrastructure to deploy

1. **Hardware Resources**
    - **Servers**: The customer opts to use their own internal servers as cloud servers. These servers have sufficient computing power and memory to meet the needs of the medical platform.

- **Storage Space**: Data storage is managed using the customer's own servers. This may involve configuring RAID arrays or something to enhance data reliability and access speed, along with backup mechanisms to prevent data loss.

2. **Software Environment**
   - **Operating System**: A popular Windows Server version is chosen as the operating system, offering a graphical interface and broad support.
   - **Python Environment**: Python and the Flask framework are installed for web application development.
3. **Network Resources**
   - **SSL Certificate**: An SSL certificate is configured to enable HTTPS encrypted communication for the website. This is particularly important for protecting the transmission of medical information.
4. **Services and Tools**
   - **CI/CD Tools**: GitHub Actions are used for continuous integration and continuous deployment, automating the testing and deployment processes to improve development efficiency and code quality.
5. **Security Measures**
   - **Access Control**: Identity verification and access control mechanisms are implemented to ensure that only authorized users and medical professionals can access the backend management interface and sensitive data.

Sprint 2

In Sprint 2, our focus is on implementing functionalities critical to the client's needs, primarily addressing user stories labeled as "Must have." with emphasizing epic of script execution and onitoring, script management and intergrate. Additionally, we'll discuss the foundational logic design for the wedsite, particularly the logic behind implementing file path changes for scripts, which is essential for our solution's architecture.

Sprint 2 planning

Sprint 2 Review

Sprint 2 Retrospective

Sprint 2 planning

Driver	@Chloe_Duan @Lingyi Kong
Approver	@yijun liu
Contributors	@Chloe_Duan @Yuncong Ji @Zhihao Liang @Junhao KONG @Lingyi Kong @yijun liu
Informed	@Wei Wang
Objective	Make plans on dev tools to be used, features to be implemented, problems that might happen, milestones to be achieved.
Due date	May 2
Key outcomes	A web interface includes login / dashboard/ script execution/ profile pages and interacts with data in database
Status	NOT STARTED / IN PROGRESS / COMPLETE

🤔 Problem Statement

As we embark on this sprint, our team faces the multifaceted challenge of integrating new technologies into our project under a tight deadline. The dual pressures of academic commitments and internships have the potential to lead to uneven workloads, with some team members at risk of becoming overburdened.

Our project structure divides the development team into two focused groups: one on frontend tasks and the other on backend tasks. While this specialization enhances the flexibility of the project, it also presents the risk of integration difficulties between the two subsystems. Synchronizing the frontend and backend components will be crucial and may require additional coordination and robust integration testing.

Additionally, the diversity of operating systems used by our team members—ranging from Windows to Mac—introduces potential compatibility issues. It is imperative that we devise a development environment and workflow that are platform-agnostic, ensuring seamless collaboration and uniformity in the deployment process across varying configurations.

The upcoming sprint will concentrate on not only advancing the project but also establishing practices that mitigate these risks. We will focus on fostering communication across the frontend and backend teams, allocating time for thorough integration testing, and standardizing our development practices to accommodate the variety of operating systems in use. By proactively addressing these concerns, we aim to maintain a balanced workload among team members and ensure the successful progression of the project.

🛠️ Prototype Design

FL -Koala

Figma

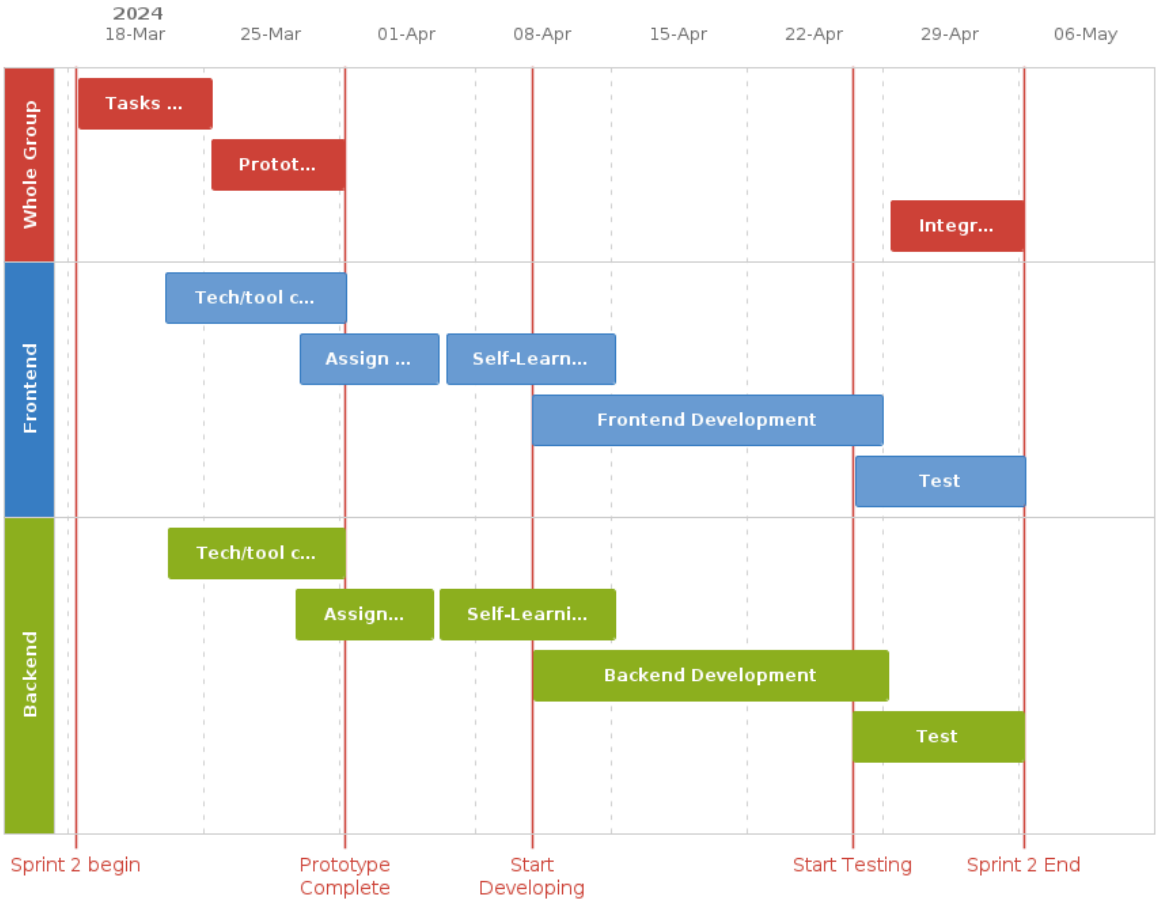
FL -Koala Edited 9 days ago

🎯 Scope

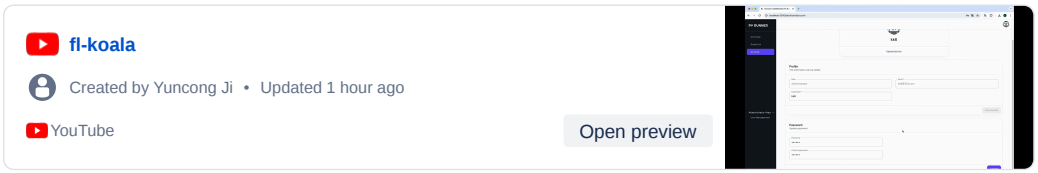
	As	I want to	So that	Task - Dev level
1	Radiation therapist	Run a specific script from a web interface easily	I can perform analysis tasks using python from any location without local installations or configurations	• Develop a backend service to execute scripts and handle session persistence for long-running tasks. • Integrate front-end interfaces with the backend to enable script execution from a web interface. • Develop a web interface for uploading, modifying, and executing scripts. • Develop automated tests for your scripts.
2	Radiation therapist	Support multiple files or folders uploading	I can analysis multiple parents information simultaneously	• Implement a file upload service on the backend that supports specific file formats and sizes. • Develop file upload functionality on the frontend. • Create a file management system to organize and store uploaded files securely.
3	Radiation therapist	Obtain the output of scripts	I can get wanted one or lots of processed patients information	• Develop a backend function to capture and store script outputs in a structured format. • Implement real-time output streaming to the frontend during script execution, including sections on the web page for success messages, error notifications, and detailed results.
4	IT department developer	Website to integrate smoothly with SQL-based databases	I can manage data consistently with other applications we use.	• Design and set up a SQL database schema that supports application needs. • Ensure data consistency and integrity across different application components.
5	Radiation therapist	Have a user-friendly script execution dashboard	I can easily find and run the script I want	• Design a dashboard that aggregates script execution statistics and history. • Implement interactive elements in the dashboard for better user engagement.
6	Radiation therapist	Access detailed usage instructions for each script directly	I can use scripts efficiently and correctly without extensive training	• Develop a context-sensitive help system that displays usage instructions based on the user's actions and roles. • Integrate instructions within the application to guide users through complex script setups.
7	Radiation therapist	Be able to save frequently used scripts	I can easily and conveniently execute the scripts I need	• Implement a feature to mark scripts as favorites and quickly access them from the dashboard. • Develop backend support for user preferences that remembers and prioritizes

				frequently used scripts.
8	Radian Researchers	version control all my scripts	I access history scripts and manage all scripts easily	• Provide a user interface for version history, diff views, and rollback options.
9	Department administrator	Have two groups of users, administrator and normal user	Coworkers with proficient coding knowledge can upload, edit and delete needed scripts and the other group can focus on script execution	• Develop a comprehensive user management system that supports two roles (Administrator & User) with different permissions. • Implement role-based access control within the application to ensure secure operations. • Create an administrator role in the web interface to oversee user management, including roles assignment, access permissions, create and delete user. • Implement features for account management, enabling users to update profiles and settings.
10	Department administrator	Let different users access the corresponding system based on their role	They can access different features based on their permission.	• Create dynamic user interfaces that adjust based on the user's role and permissions. • Ensure that API endpoints enforce role-based access controls strictly.
11	Department administrator	Have a role of administrator for user account management	The efficient management of users can be handles within teams.	• Implement administrative functions to manage user roles, permissions, and access controls.

🗓️ Timeline



Project Demonstration



https://www.youtube.com/watch?v=pmYtdWO1FxU

Milestones and deadlines

Milestone	Owner	Deadline	Status
Finalise dev tool selection.	@Chloe_Duan	Apr 1, 2024	COMPLETE
Assign tasks for frontend and backend.	@Chloe_Duan @Zhihao Liang	Apr 1, 2024	COMPLETE
Finish prototype development.	@Lingyi Kong @yijun liu	Apr 1, 2024	COMPLETE
Frontend and backend development.	Dev team	Apr 28, 2024	COMPLETE

Complete frontend testing	@Chloe_Duan @Junhao KONG @yijun liu	Apr 28, 2024	COMPLETE
Complete backend testing	@Zhihao Liang @Yuncong Ji @Lingyi Kong	May 1, 2024	COMPLETE
Complete Integration testing.	Dev team	May 2, 2024	COMPLETE
Finalise all sprint documentations.	Dev team	May 2, 2024	COMPLETE
Finalize all sprint tasks	Dev team	May 2, 2024	COMPLETE

🔗 Reference materials

Sprint 2 Review

Duration	Mar 23, 2024 to May 2, 2024
Goal	Demonstrate and review the work completed during the Sprint.
Team	FL-Koala
Participants	@Chloe_Duan @Zhihao Liang @yijun liu @Yuncong Ji @Junhao KONG @Lingyi Kong
Informed	@Wei Wang
Agenda	Sprint Goal Sprint Deliverables Incomplete works Challenges Performance Review • Burndown Chart • Performance Analysis • Actions to do Demonstration Next Steps Plan Feedback and Discussion

🏁 Sprint Goal

In this Sprint, our objective is to build and implement the essential user stories that establish the core features of our online script execution platform. By adopting a frontend-backend separation approach, we aim to finalize the login, dashboard, user profiles, script execution, and script overview functionalities. Our focus is to ensure smooth data access and manipulation from the database, as well as reliable online script execution capabilities for our users.

🎯 Sprint Deliverables

- We developed a feature that allows administrators to upload scripts in zip file formats, ensuring a seamless setup and preparation for script execution without manual file management.
- We implemented a user-friendly web interface that enables users to execute specific scripts directly from the web, bypassing the need for local Python installations or command-line interactions.
- We enabled real-time monitoring of script execution, allowing users to view/download outputs instantly, which helps in quick decision-making and troubleshooting.
- A secure login system was established to manage user authentication and script operations securely, providing role-based access control to ensure that only authorized users can perform sensitive operations.

- We provided administrative tools for managing user roles and permissions, streamlining the process of user management within teams and ensuring system integrity.
- The interface was enhanced to adapt to various output types, facilitating users in their studies by allowing them to analyze script outputs directly without needing additional software tools.
- A dashboard was developed to give users real-time data on script status, enhancing transparency and operational oversight.
- We integrated a CI/CD pipeline into our GitHub repository to automate the testing and deployment processes, ensuring that all script updates are smoothly transitioned into production.
- User stories included developments for easy script uploading, deleting, and saving functionalities, enhancing the overall user experience and maintaining the security and integrity of the platform.

🗒️ Incomplete Work

While we made substantial progress this sprint, certain enhancements remain incomplete but are crucial for improving user interaction with our platform. Our next steps include refining the web interface to enhance clarity and user-friendliness. Specifically, we aim to implement a dynamic script information overview on the homepage, which will display newly updated scripts, the most popular scripts, and those recently modified. This feature will allow users to quickly access and assess scripts based on current relevance and community engagement.

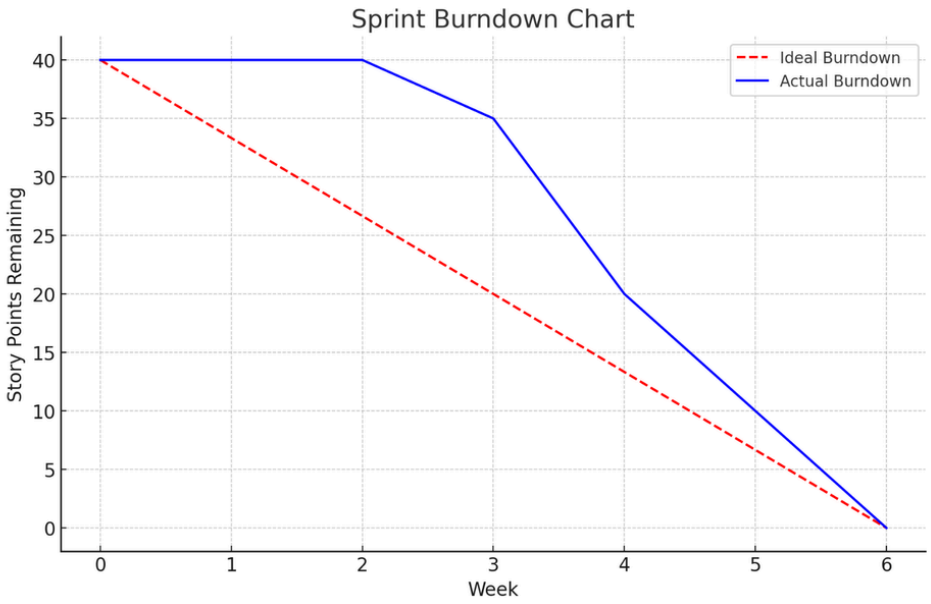
Additionally, we plan to integrate functionality to allow users to favorite or unfavorite scripts. This will streamline the user experience, enabling one-click operations to customize their interface and interact more efficiently with the scripts they need most often. These features are designed to not only personalize the user experience but also to enhance the accessibility and efficiency of the platform, making it more intuitive for all users.

⛰️ Challenges

Throughout this Sprint, we faced several challenges that tested our adaptability and resilience. Balancing our full-time studies with professional internships and side jobs, our team managed the complex task load with varying degrees of success. Initially, the frontend group had to scale a steep learning curve to familiarize themselves with the new technologies that underpin our development. Additionally, diverse operating systems and development tools across team members' laptops presented integration hurdles. Varied progress in frontend and backend tasks also led to synchronization difficulties, at times resulting in backend development outpacing frontend efforts.

⏮️ Performance Review

- **Burndown Chart**



- **Performance Analysis**

The team's performance depicted in the Burndown Chart highlights a delayed start, with no story points completed in the first two weeks, indicating a potential struggle with either task initiation or encountering early impediments. However, the performance surged in week 4 and continued steadily in weeks 5 and 6, culminating in the completion of all story points by the end of the sprint. This pattern suggests a period of adjustment followed by accelerated productivity, with the team demonstrating an ability to catch up and fulfill sprint commitments despite initial delays.

- **Action to do**

1. Conduct a thorough review of sprint planning and task breakdown to ensure tasks are actionable from the outset.
2. Address any obstacles or complexities early on, possibly through more focused problem-solving sessions or clearer prioritisation.
3. Reinforce the importance of maintaining a consistent work pace throughout the sprint to avoid a heavy workload towards the end.
4. Recognise and analyse successful strategies and practices that led to the eventual completion of goals to replicate these in future sprints.
5. Continue to celebrate the team's successes and learn from each sprint's unique challenges and outcomes.

💻 Demonstrations

⏭️ Next Steps Plan

For the upcoming Sprint, we plan to continue our progress by completing the remaining 'Should-have' and 'Could-have' user stories, thus enriching our web service's functionality. Furthermore, we will refine the user interface to enhance its aesthetics and user experience. By the conclusion of the next Sprint, we intend to integrate all components cohesively and execute a comprehensive integration test. This will ensure that our application operates seamlessly across different user interactions.

😀 Feedback and Proposals

Our clients did not attend the sprint review meeting as they went on vacation and will not come back to work until `May 13, 2024`. Therefore, we did not have chance to show our work to them and get feedbacks from their at the end of this sprint. We plan to schedule a meeting with the clients to show them our deliverables in sprint 2 and ask for feedbacks on how to adjust our web interface.

However, we did take a meeting inner our group for sprint review, and each team members expressed their opinions to make it more smooth and opportune to complete the upcoming sprint 3.

- There was a shared agreement on the challenges faced during the integration of frontend and backend systems. The team discussed the need for better synchronisation mechanisms, as highlighted in the sprint planning documentation, which could prevent potential delays and ensure smoother integration in future sprints.
- Feedback from the frontend team emphasized the steep learning curve for new technologies, which was also noted in the sprint retrospective. Participants appreciated the gradual improvement in handling these technologies towards the end of the sprint. The need for additional training sessions and resources was suggested to accelerate the learning process and reduce the initial lag in productivity.
- As documented in the sprint planning, the diverse operating systems used by the team members posed significant compatibility challenges. The discussion underscored the necessity of establishing a protocol for regular compatibility checks, which was a proposed action item in the retrospective to prevent such issues in the future.
- The enhancements made to the user interface and script management functionalities were well-received, as seen in the sprint review documentation. However, feedback also pointed towards the need for continuous improvement in the UI to cater to the evolving needs of the users, particularly in simplifying script management tasks.

Sprint 2 Retrospective

📋 Overview

Date	May 1, 2024
Team	FL-Koala
Participants	@Chloe_Duan @Zhihao Liang @Lingyi Kong @yijun liu @Yuncong Ji @Junhao KONG

💭 Retrospective

Start doing	Stop doing	Keep doing
• Enhance cross-team communication between frontend and backend groups.	• Cease the practice of postponing task initiation.	• Holding regular meetings.
• Allocate adequate time for task completion.	• Avoid overlooking or dismissing suggestions.	• Offer mutual support among team members.
• All members actively contribute to the project's success.	• Eliminate casual attitudes toward development.	• Maintain in-group communication.
• Identify problems early and address issues proactively.	• Refrain from disconnecting from team activities.	• Follow the scrum development process

✅ Action items

Actions	Owner
• Cross-team weekly Sync-up: Initiate a weekly sync-up meeting between the frontend and backend teams to ensure alignment and early identification of integration issues. Set up the initial schedule and invite all necessary team members.	@Lingyi Kong
• Task Start Timeline: Develop a task start timeline to prevent the late initiation of tasks. Integrate this into the project management tool by the next sprint planning session, ensuring tasks have clear start dates.	@Junhao KONG @yijun liu
• Idea Review Sessions: Schedule regular idea review sessions to ensure all team member suggestions are heard and evaluated. This can be included as part of the regular meetings. Incorporate these sessions into the existing meeting structure.	@Chloe_Duan
• Development Standards Workshop: Conduct a workshop to reinforce development standards and the importance of maintaining a professional	@Zhihao Liang

approach to work. Organise the workshop and prepare relevant materials.	
• Issue Tracking System: Implement or improve the use of an issue tracking system to ensure problems are reported, tracked, and resolved in a timely manner. Select an appropriate tool and set up a process for the team to follow.	@Yuncong Ji
• Compatibility Check Protocol: Establish a protocol for regular compatibility checks across different operating systems used by the team to catch and resolve issues early. Create a checklist and schedule for these checks, ensuring that every piece of code is tested on both Windows and Mac environments before being merged.	@Zhihao Liang

Sprint 3

In Sprint 2, we successfully implemented all 'must have' user stories but identified several logical bugs impacting performance and user experience. Sprint 3 will focus on resolving these issues to stabilize and refine our application. Additionally, we will implement selected 'non-must have' user stories from our product backlog to enhance functionality. Key improvements will include detailed script information and the capability for batch script processing, aiming to improve user efficiency and interface intuitiveness. This sprint is critical for transitioning our system from functional to polished, ready for broader deployment and use.

Sprint 3 planning

Meeting notes

Date	Meeting notes	Note
Mar 15, 2024	2024-03-15 Client meeting notes	
Mar 18, 2024	2024-03-18 Internal meeting notes	
Mar 20, 2024	2024-03-20 Stand up meeting with Wei	
Mar 21, 2024	2024-03-21 Inner-Group Assignment Discussion	
Mar 24, 2024	2024-03-24 Sprint 1 Review	
Apr 10, 2024	2024-04-10 Stand up meeting with Wei	
Apr 10, 2024	2024-04-10 Client meeting	
Apr 17, 2024	2024-04-17 Stand up meeting with Mingye	
Apr 24, 2024	2024-04-24 Stand up meeting with Mingye	
May 1, 2024	2024-05-01 Stand up meeting with Wei	

2024-03-15 Client meeting notes

Meeting overview

- Meet client for fitst time. Introduce team. Get project context and client needs.
- Key attendees: clients.
- Processes: Introduce with each other; Clients explain their needs; Q&A.

Meeting minutes

Date	Attendees	Notes, decisions and action items
Mar 15, 2024	All	Introduction: • Leah is looking to create a webpage to run scripts that automate various tasks in the hospital radiotherapy department • Many scripts have been created by different people to save time by automating manual processes • Goal is to have a webpage where staff can easily run scripts, even if they are not familiar with coding • If script doesn't work, physicist can look at code, fix it, and update on the webpage Webpage Requirements: • Create a webpage using a framework like Flask (open to other suggestions) • Nice design with categories/groups for scripts • Scripts will require various user inputs and produce different outputs • Need to handle running scripts in the background • Hosted on a university server initially, later deployed to hospital servers • Low expected traffic, a couple users per day Additional Tasks: • Alongside the main webpage and repository, individual team members can choose additional small scripting tasks to automate other radiotherapy processes • Examples: Averaging 3 scanned films, offline checking of patient treatment margins • Leah to provide input data, expected output, and context for each task • Scripts work with relatively small file sizes, at most a couple GB at a time Other Notes:

		• Complete freedom given on webpage design, just keep it simple and not too busy • Documentation needed at 3 levels: 1. For end users on how to use each script (content from Leah) 2. How the GitHub repo and webpage is set up 3. Code comments in modified scripts • Some security required to limit script editing to approved list of staff • No need to anonymize patient data, Leah will provide pre-anonymized sample data • No major speed requirements for most scripts, open to optimizing a couple slow ones Next Steps: • Leah to cleanup requirements doc specifying must-haves vs nice-to-haves

Open action items

- [x] Team to setup Google Drive for document sharing
- [x] Schedule recurring fortnightly meetings, with Leah mainly coordinating with Kelvin and Lynn
- [x] Team members to discuss splitting up tasks based on webpage, repository, scripting preferences

2024-03-18 Internal meeting notes

Meeting overview

- Meeting goal: Internal feedback after clients' meeting; Decide what to do next.
- Key attendees: All team member.
- Processes: Discussion; brainstorm.

Meeting minutes

Date	Attendees	Notes, decisions and action items
Mar 18, 2024	@Zhihao Liang @Lingyi Kong @Chloe_Duan @Yuncong Ji @Junhao KONG @yijun liu	1. Discuss assignment 1 2. Select a user story template 3. brainstorm user cases 4. Finalize the product backlog following the client meeting 5. Create a Trello workspace 6. Distribute Assignment 1 tasks among team members

Action Notes highlight

- [x] Background description, client goals, motivation @Junhao KONG @yijun liu
- [x] Analysis of requirements (User Stories or Use Cases) & Plan @Yuncong Ji @Chloe_Duan
- [x] Development environment @Lingyi Kong
- [x] Meetings & GitHub @Zhihao Liang

Decision

Product backblog template

2024-03-20 Stand up meeting with Wei

Meeting overview

- Meeting goal: Update each other's work; get feedback from supervisor.
- Key attendees: All team member.
- Processes: Stand-up meeting.

Meeting minutes

Date	Attendees	Notes, decisions and action items
Mar 20, 2024	@Zhihao Liang @Lingyi Kong @Chloe_Duan @Yuncong Ji @Junhao KONG @yijun liu @Wei Wang	1. Each member update their progress a. @Chloe_Duan and @Yuncong Ji are still working on requirement analysis b. @Zhihao Liang has initialize github repo and make a meeting note for Mar 18, 2024 meeting c. @Junhao KONG and @yijun liu showed the progress of background description and client goal and motivation and wei suggested that those part need to be revised according to workshop material. d. @Lingyi Kong is working on project backlog review. 2. Advice from Wei: a. Progress Transparency b. PO should allocate some responsibility to other team members c. Keep updating progress

Action items highlight

- [x] Allocate responsibity to other member @Chloe_Duan

2024-03-21 Inner-Group Assignment Discussion

Meeting

- Meeting goal: Check each other’s work; solve problems.
- Key attendees: All team member.
- Processes: Discussion; brainstorm.

Meeting minutes

Date	Attendees	Notes, decisions and action items
2024-03-21	@Zhihao Liang @Lingyi Kong @Chloe_Duan @Yuncong Ji @Junhao KONG @yijun liu	1. Select tool and template for Persona and Do/Be/Feel list. 2. Cooperate on creating Personas, Do/Be/Feel list and Goal Model. 3. Brainstorm on requirements based on project summary. 4. Discuss on possible implementation methods of some requirements. 5. Set a time for the next meeting.

Action items highlight

- [x] Finish user story @Chloe_Duan @Yuncong Ji

Decision

Using Miro template for Persona and DO-BE-FEEL working board

2024-03-24 Sprint 1 Review

Meeting overview

- Meeting goal: Finalize sprint 1 content.
- Key attendees: All team members.
- Processes: Discussion about product backlog; overview whole content.

Meeting minutes

Date	Attendees	Notes, decisions and action items
2024-03-24	@Zhihao Liang @Lingyi Kong @Chloe_Duan @Yuncong Ji @Junhao KONG @yijun liu	1. Continue working on requirements (user stories), and assign priority and size estimation to each requirements. 2. Create product backlogs on Trello and distribute backlogs to Sprint 2 and Sprint 3 according to priority and estimated story points. 3. Organise materials on Confluence. 4. Tick completed items on Checklist. 5. Structure the Github repository, update Readme file, and generate Release tag. 6. Final check and ready to submit the assignment.

Action items highlight

- [x] Export PDF from confluence. @Chloe_Duan
- [x] Upload PDF to Github. @Zhihao Liang

Decision

- Go through each user story and decide for priority and size estimation
- Spilt task for sprint 2 and sprint 3

2024-04-10 Stand up meeting with Wei

Meeting overview

- Meeting goals: Ask questions and discuss functions that the team is unsure about.
- Key attendees: All team members and Wei.
- Processes: The development of script running and user management functions is discussed.

📝 Meeting minutes

Date	Attendees	Notes, decisions and action items
Apr 10, 2024	@Zhihao Liang @yijun liu @Lingyi Kong @Chloe_Duan @Yuncong Ji @Junhao KONG @Wei Wang	1. Split the task for Sprint 2 into different team members. 2. Organise materials on Confluence. 3. The front end team started learning the react framework and started trying to write code. 4. The back-end team started building the database and experimenting with the back-end code.

✅ Open action items

- [x] The front-end team configured the code environment.
- [x] The back-end team set up the database.
- [x] Team members to discuss splitting up tasks based on webpage.

2024-04-10 Client meeting

Meeting overview

- Meeting goals: Determine important issues such as script path and user permissions with the client.
- Key attendees: Client and all team members.
- Processes: We discussed the problems with code implementation and showed the client our initial Figma UI design for the web, which they were very pleased with and suggested areas for improvement.

Meeting minutes

Date	Attendees	Notes, decisions and action items
Apr 10, 2024	@Zhihao Liang @Lingyi Kong @Chloe_Duan @Yuncong Ji @yijun liu @Junhao KONG	1. Based on the client's satisfaction with the UI design, we decided to start implementing the front end of the web using React based on the design of Figma. 2. We decided that the team responsible for the front-end code was: @Chloe_Duan @yijun liu @Junhao KONG and the back-end team should have: @Yuncong Ji @Lingyi Kong @Zhihao Liang 3. We decided to separate the front and back ends, and the team members of the two sections would have regular meetings to communicate.

Open action items

- [x] The front and back end team members meet regularly or zoom meetings respectively.
- [x] Confluence and Trello provided regular updates on our progress.

2024-04-17 Stand up meeting with Mingye

Meeting overview

- Meeting Goal: Check the current process and sprint 2 progress.
- Key attendees: Mingye Li, all team members.
- Processes: Meet the new supervisor Mingye Li for first time, and he advised us on our current process. We discussed about formatting issues with Confluence and Trello, as well as issues with user stories.

Meeting minutes

Date	Attendees	Notes, decisions and action items
Apr 17, 2024	@Chloe_Duan @Lingyi Kong @Zhihao Liang @yijun liu @Junhao KONG @Yuncong Ji	• Discussion on the structure and handling of user stories, backlogs, and testing methodologies. • Considerations for API security, specifically around using session cookies or JWT for authentication. • Feedback on team contributions, emphasizing the need for improvement in areas such as documentation, test case development, and the use of Confluence for organizing content conceptually rather than sprint-based. • Advice on UI design principles and the importance of error handling and user feedback mechanisms in the application. • Suggestions on how to improve the GitHub repository with detailed change logs and handover instructions. **Decisions** • Consolidation and reclassification of user stories to ensure clarity and manageability within the project management tools. • Agreement on using JWT for session management, emphasizing the simplicity and efficiency for the current project needs. • Enhancement of UI design following recognized usability principles to ensure the application is intuitive and efficient for users.

Action items highlight

- [x] Frontend code @yijun liu @Chloe_Duan @Junhao KONG
- [x] Backend code @Lingyi Kong @Zhihao Liang @Yuncong Ji
- [x] **Subsequent integration:** and modification of front-end code @Chloe_Duan @Zhihao Liang

- [x] **Enhance UI Design:** Apply the ten usability principles discussed to improve the user interface design, ensuring consistency and efficiency across the application. @yijun liu
- [x] **Improve GitHub Documentation:** Update the GitHub repository to include a change log and detailed setup and handover instructions to assist future developers or the client in understanding and using the project. @Yuncong Ji
- [x] **Confluence Documentation Strategy:** Restructure Confluence documentation to be concept-based rather than sprint-based to enhance the organization and accessibility of project information. @All
- [x] **Session Management Implementation:** Implement JWT for session management and ensure proper integration and security testing. @Lingyi Kong
- [x] **Prepare for Client Meetings:** Ensure readiness for upcoming client meetings by preparing demonstrations that showcase the progress and functionality of the application, particularly focusing on robustness and user experience. @ All

2024-04-24 Stand up meeting with Mingye

Meeting overview

- Meeting goals: Improve the Confluence page content.
- Key attendees: All team members and Mingye Li
- Processes: We discussed the current project progress and where Confluence needs to change.

📝 Meeting minutes

Date	Attendees	Notes, decisions and action items
Apr 24, 2024	@Zhihao Liang @yijun liu @Lingyi Kong @Chloe_Duan @Yuncong Ji @Junhao KONG	**Meeting Notes** • Discussion about the project's current status including challenges with integration of the front end and back end, setup and management of GitHub branches, and code review practices. • Detailed discussions on testing strategies, including acceptance and integration testing, with a focus on improving documentation with screenshots and tester details for accountability. • Clarifications provided on project scope, necessary diagrams for documentation, and proper utilization of Confluence for organizing project information. • Advice on job hunting and improving employability with insights into the IT job market and the importance of internships and certifications. **Decisions** • Decision to continue development locally, with the client responsible for deployment on their hospital server. Clarifications from the client will be sought regarding the specifics of the deployment environment. • Emphasis on documenting the scope within the project's lifecycle, especially what is within the scope for the upcoming sprints and what isn't. • Commitment to enhancing UI design incrementally, ensuring each component is completed and tested thoroughly before moving to the next. • Approval to maintain current GitHub branch structure for development convenience but to simplify it before the project's conclusion.

✅ Open action items

- [x] **API**: connection issues resolved by @Chloe_Duan @Zhihao Liang

- [x] **Login page**: @Chloe_Duan ; Script List and Overview page @yijun liu ; User Management and Script Execution page @Junhao KONG
- [x] **Clarify Deployment Details:** Directly inquire with the client about the specifics of the server where the project will be deployed to ensure compatibility and proper setup in project documentation. @Zhihao Liang
- [x] **Documentation Improvement:** Ensure all test cases include detailed information, such as screenshots and tester details. Incorporate a proper structure in Confluence for documenting code reviews and project scope. @Zhihao Liang
- [x] **Diagram Inclusion:** Include necessary diagrams like ER diagrams for database structure and sequential diagrams for use cases to clarify project architecture and workflow. @Chloe_Duan
- [x] **Code Review Documentation:** Set up a Confluence page dedicated to documenting the code review process, including a checklist and responsibilities. @Yuncong Ji
- [x] **Prepare for Finalization:** Plan to simplify the GitHub repository structure towards the project's end, merging significant changes into fewer branches and preparing for a clean deployment. @Lingyi Kong

Daily Stand-up Meeting for Sprint 2

ℹ️ All team members should provide their priorities, progress, and problems twice a week in this report.

Team name	FL-Koala
Direct supervisor	@Chloe_Duan
Table of contents	

🗓️ Friday <23 April>

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Chloe_Duan	1. Learning React and Typescript 2. Implement login page	Completed	1. Had issues with solutions of authorization 2. No idea how to connect backend APIs
2	@yijun liu	1. Learning React and Typescript 2. Finish script list page	Completed	1. Confused about JSON data format from the server. 2. Had issues with running the backend project
3	@Junhao KONG	1. Learning React and Typescript 2. Code for User management page	Completed	1. Not easy-to-understand template code 2. Had issues with connecting the user database
4	@Yuncong Ji	1. Learning Flask and Python 2. Design database schema and associated apis 3. Finish script module in Flask	Completed	1. Had difficulty in running the backend project 2. Database integration challenges
5	@Zhihao Liang	1. Implement authorization	Completed	No

		module 2. Set up cloud database		
6	@Lingyi Kong	1. Code for Favorite backend 2. Design APIs 3. Learning Flask and Python	Completed	No

🗓️ Thursday <25 April>

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Chloe_Duan	1. Debug broken issue 2. Script List api connect 3. integration three page 4. forget password	Completed	1. Inactive backend team members 2. Data format confusion about script list 3. URL routing in Next.js
2	@yijun liu	1. Dashboard UI design 2. Script list filter 3. Script list running button	Completed	1. Responsive design compatibility issues
3	@Junhao KONG	1. Script running page 2. Fix layout errors	Completed	1. User information loss 2. Steep learning curve about next.js
4	@Yuncong Ji	1. UML image generation 2. Refactor previous code	Completed	No
5	@Zhihao Liang	Code for user management API	Completed	No
6	@Lingyi Kong	Code for authentication backend	Completed	No

🗓️ Tuesday <27 April>

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Chloe_Duan	Connect the APIs for frontend and backend	Completed	1. Incomplete input from some team members

2	@yijun liu	Code for script overview page	Completed	1. Configuration issues with deployment scripts
3	@Junhao KONG	Code for script execution page	Completed	1. Complexity in handling user permission
4	@Yuncong Ji	Code for script backend and database	Completed	No
5	@Zhihao Liang	Code for script execution	Completed	No
6	@Lingyi Kong	Code for authentication backend	Completed	No

🗓️ Monday <30 April>

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Chloe_Duan	1. Connect the frontend and backend APIs 2. Merge frontend branches	Completed	1. Confused about setState in Reactjs 2. Had issues with inserting image
2	@yijun liu	Edit Confluence meeting minutes and code review as well as test case	Completed	No
3	@Junhao KONG	Edit Confluence code review and test cases	Completed	No
4	@Yuncong Ji	Code for backend and work for confluence, website test	Completed	No
5	@Zhihao Liang	Code for refactoring existing backend code	Completed	No
6	@Lingyi Kong	Edit Confluence retrospectives, code review and test cases	Completed	No

Test

Sprint 2 Testing

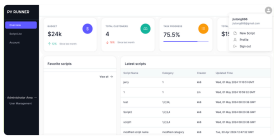
- Test case 01
- Test case 02
- Test case 03
- Test case 04
- Test case 05
- Test case 06

Sprint 2 Testing

- Test case 01
- Test case 02
- Test case 03
- Test case 04
- Test case 05
- Test case 06

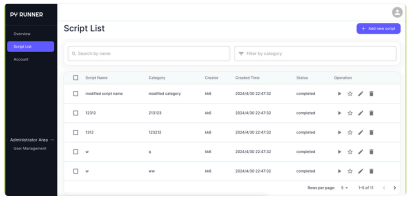
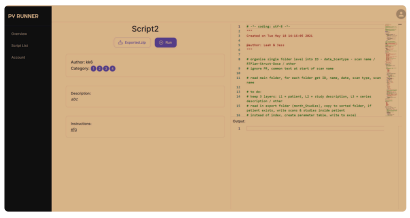
Test case 01

Test Case ID	1
Created By	@Zhihao Liang
Test Case Description	Log in to the system
Tester's Name	@Zhihao Liang
Date Tested	Apr 28, 2024
Test Case (Pass/Fail/Not Executed)	Pass

Step #	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:3000	Site should open	As Expected		Pass
2	Enter email and password email:jiutong666@gmail.com password:123456	Display email and password with placeholder	As Expected		Pass
3	Click login	Go to the dashboard page	As Expected		Pass

Test case 02

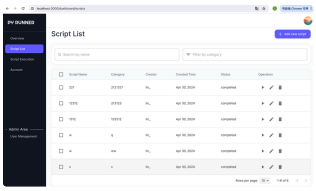
Test Case ID	2
Created By	@Zhihao Liang
Test Case Description	Execute a script from script management page
Tester's Name	@Zhihao Liang
Date Tested	Apr 30, 2024
Test Case (Pass/Fail/Not Executed)	Pass

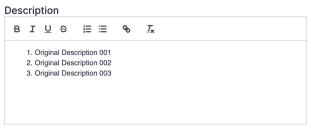
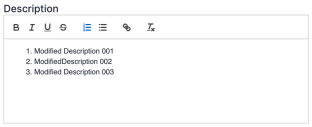
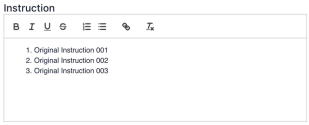
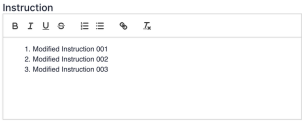
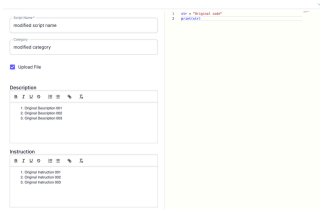
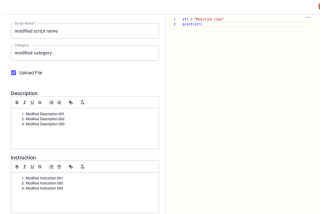
Step #	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:3000/dashboard/scripts	Script management page should open	As Expected		Pass
2	Click "Run Script" button in a row in the action column	Redirect to /dashboard/scripts/{scriptid}/execution	As Expected		Pass
3	If script requires file upload, upload a valid zip file	Upload input should accept file and display uploaded	As Expected		Pass
4	Click "Run" button	If upload required, execution starts after upload. Without upload,	As Expected		Pass

		execution starts immediately.			
5	View execution log in output area	Execution log should display progress and results	As Expected		Pass
6	If available, download output file	Download should start and file should be correct	As Expected		Pass

Test case 03

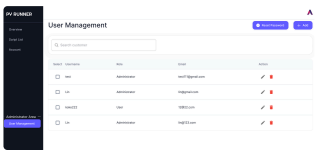
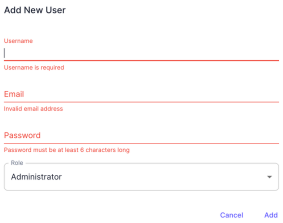
Test Case ID	3
Created By	@Lingyi Kong
Test Case Description	List available scripts and modifications on script details
Tester's Name	@Lingyi Kong
Date Tested	Apr 30, 2024
Test Case (Pass/Fail/Not Executed)	Pass

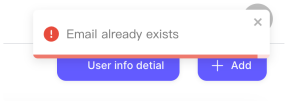
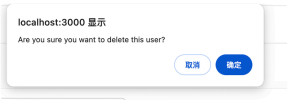
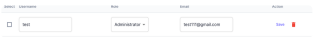
Step #	Owner	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	@Lingyi Kong	Navigate to the 'Script List' page.	Show all scripts in the 'Script List' page.	As expected		Pass
2	@Lingyi Kong	Click the edit button of the script to be modified, it will redirect to the script content page; Replace content in "Script Name" input box with desired value; Click "submit" to submit modification.	'Script name' be modified successfully and be stored in the database.	As expected	before: after: 	Pass
3	@Lingyi Kong	Click the edit button of the script to be modified, it will redirect to the script content page; Replace content in "Category" input box with desired value;	'Category' be modified successfully and be stored in the database.	As expected	before: after: 	Pass

		Click “submit” to submit modification.				
4	@Lingyi Kong	Click the edit button of the script to be modified, it will redirect to the script content page; Replace content in “Description” input box with desired value; Click “submit” to submit modification.	‘Description’ be modified successfully and be stored in the database.	As expected	before:  after:  	Pass
5	@Lingyi Kong	Click the edit button of the script to be modified, it will redirect to the script content page; Replace content in “Instruction” input box with desired value; Click “submit” to submit modification.	‘Instruction’ be modified successfully and be stored in the database.	As expected	before:  after:  	Pass
6	@Lingyi Kong	Click the edit button of the script to be modified, it will redirect to the script content page; Replace content in right hand side input box with desired value; Click “submit” to submit modification.	‘Code’ be modified successfully and be stored in the database.	As expected	before:  after: 	Pass

Test case 04

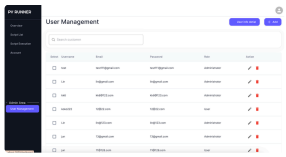
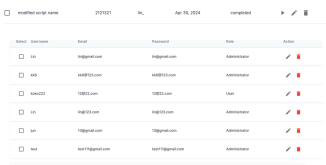
Test Case ID	4
Created By	@yijun liu
Test Case Description	Manage user information from the User Management page
Tester's Name	@yijun liu
Date Tested	Apr 30, 2024
Prerequisite	User must be logged in as administrator
Test Case (Pass/Fail/Not Executed)	Pass

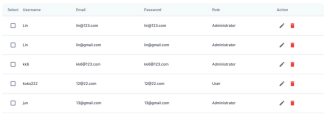
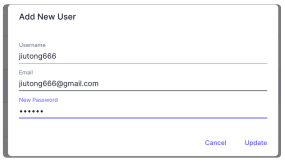
Step #	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:3000/dashboard/customers	User management page should open	As Expected		Pass
2	Click "Add" button in the upper right corner	The user information will be added after entering user information and selecting an identity	As Expected		Pass
		If the added information is empty, the information cannot be added	As Expected		
		If the added user email address already exists,	As Expected		Pass

		a message will be displayed			
3	If a specific user needs to be searched, enter the user name in the search bar to search for the specific user	Upload input should accept file and display uploaded	As Expected		Pass
4	If a user needs to be deleted，click the delete button	A pop-up window will pop up asking if you are sure to delete the user	As Expected		Pass
5	If a user's information needs to be edited or changed, click the edit button	User information can be changed directly	As Expected		Pass

Test case 05

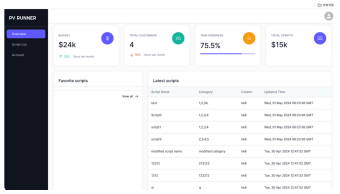
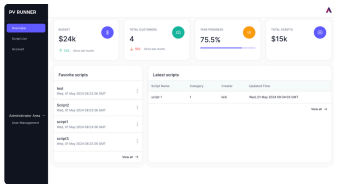
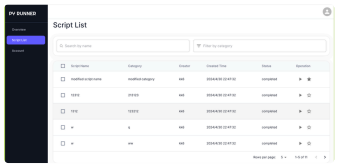
Test Case ID	5
Created By	@Yuncong Ji
Test Case Description	user account info
Tester's Name	@Yuncong Ji
Date Tested	Apr 30, 2024
Test Case (Pass/Fail/Not Executed)	Pass

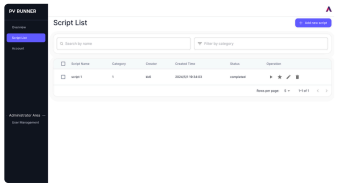
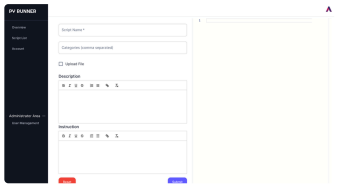
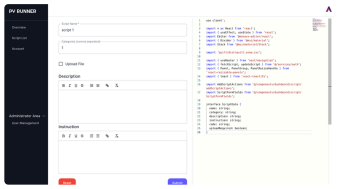
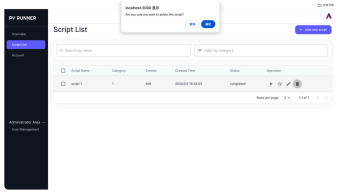
Step #	Owner	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	@Yuncong Ji	Navigate to the 'User Management' page.	Show all users in the ' User Management'' page.	As expected		Pass
2	@Yuncong Ji	Click the delete button of the user to be modified,	'user' be deleted successfully and updated in the database.	As expected	before:  after 	Pass
3	@Yuncong Ji	Click the edit button of the user to be modified Update Username,Email Password,Role	user information be modified successfully and be stored in the database.	As expected	before: after:	Pass

		Click save button to save user information				
4	@Yuncong Ji	Click “Add“ button to add new User of administrator Enter username, email, password and select user or administrator. Click “add“ to add user on page. 	“User“ be added successfully and be stored in the database.	As expected	before:  after: 	Pass
5	@Yuncong Ji	Must select button to choose user for update password click “user info detail“ to updated password click update button to save new password	‘password’ be modified successfully and be stored in the database.	As expected		Pass

Test case 06

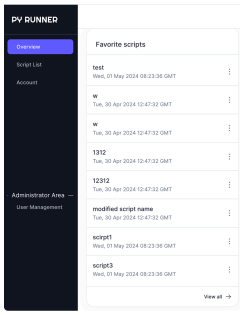
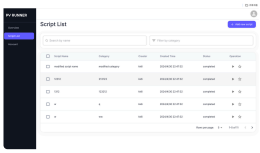
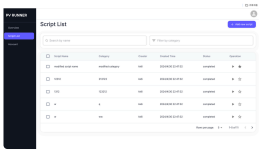
Test Case ID	6
Created By	@Junhao KONG
Test Case Description	Test different role functions (admin & user)
Tester's Name	@Junhao KONG
Date Tested	May 1, 2024
Test Case (Pass/Fail/Not Executed)	Pass

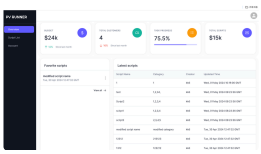
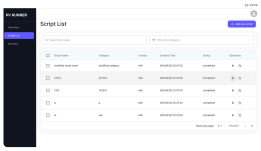
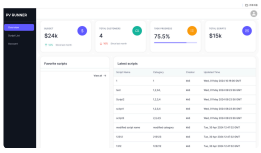
Step #	Owner	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	@Junhao KONG	Use the user account to log in. email: test111@gmail.com password:123456	Show the user page. There is no user management page.	As expected		Pass
2	@Junhao KONG	Use the admin account to log in email: kk6@123.com passowrd:123456	Show the admin page. There is a user management page.	As expected		Pass
3	@Junhao KONG	Navigate the script list page using the user account. email: test111@gmail.com password:123456	Users can not add new scripts, modify scripts, or delete scripts.	As expected		Pass

4	@Junhao KONG	Navigate the script list page using the admin account. email: kk6@123.com passowrd:123456	There is an add new script button for the admin to add new scripts. There are modify and delete buttons in the operation list.	As expected		Pass
5	@Junhao KONG	Click the Add New Scripts button using the admin account. email: kk6@123.com passowrd:123456	The add new scripts page can run successfully.	As expected		Pass
6	@Junhao KONG	Click the Modify button using the admin account. email: kk6@123.com passowrd:123456	The modify scripts page can run successfully.	As expected		Pass
7	@Junhao KONG	Click the Modify button using the admin account. email: kk6@123.com passowrd:123456	The scripts can be deleted successfully.	As expected		Pass

Test case 07

Test Case ID	7
Created By	@Junhao KONG
Test Case Description	Test favourite function script in the overview page
Tester's Name	@Junhao KONG
Date Tested	May 1, 2024
Test Case (Pass/Fail/Not Executed)	Pass

Step #	Owner	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	@Junhao KONG	Navigate to the overview page	There is a favourite scripts area in the left corner of the page.	As expected		Pass
2	@Junhao KONG	Navigate to the scripts list	There is a star button (for favourite scripts) in the operation list.	As expected		Pass
3	@Junhao KONG	Click the star button for the script called(modified script name), and navigate to the overview page.	The script called(modified script name) is shown in the favourite scripts list.	As expected		Pass

						
4	@Junhao KONG	Click the star button for the script called(modified script name) again to remove the script, and navigate to the overview page.	The script called(modified script name) is disappeared in the favourite scripts list.	As expected	 	Pass

Code Review

This document outlines the code review process employed during our project. The review was conducted manually by team members, with each member participating in an average of 3-4 reviews. This document aims to standardize our approach to ensure consistency and quality across our codebase.

Objectives

- **Identify and resolve logical errors**: Ensuring the code functions correctly according to specifications.
- **Enforce coding standards**: Maintaining a consistent style and adhering to best practices to make the code more readable and maintainable.
- **Promote code simplicity**: Encouraging clear and concise code that is easy to read and understand.
- **Evaluate reusability**: Assessing the code for potential reuse to reduce redundancy and improve efficiency.

Process

1. **Initial Review**: Each team member is assigned code segments to review based on the criteria outlined.
2. **Feedback Submission**: Reviewers provide feedback and suggest improvements through inline comments or summary reports.
3. **Implementation of Changes**: Developers address the feedback by making the necessary revisions to their code.
4. **Follow-Up Review**: Revised code segments are submitted for a second review to ensure all concerns are addressed.
5. **Approval for Pull Request**: Once the code passes the second review, it is approved to be merged into the main branch via a pull request.

Review Criteria

- **Logical Errors**: Reviewers are tasked with identifying any flaws in logic that could lead to incorrect program behavior.
- **Code Standards**: Compliance with predefined coding guidelines is checked rigorously.
- **Simplicity**: Efforts are made to keep the codebase simple and understandable.
- **Reusability**: Code is evaluated on its potential for reuse in other parts of the project or future projects.

Code Review for Scripts-related API

Project Name/Module Name: Scripts-related APIs and services

Author: @Yuncong Ji

Reviewer: @Lingyi Kong

Review Date: Apr 23, 2024 , Apr 30, 2024

Code Review for First Version

Selection Criteria:

- Priority was given to reviewing new features added in the sprint.
- Areas with high bug rates in previous releases were targeted.

Process:

- The review was initiated by examining the commits made last week, focusing on modules related to script management.
- The session was conducted remotely via a scheduled Zoom call, where each line of code was shared and discussed live.
- Feedback was documented directly in the code using comments and later summarized on this Confluence page.

Feedback Received:

1. Exception Handling: Inconsistencies in how exceptions are caught and handled.
2. Security Concerns: Potential security risks due to improper validation of script deletion logic.
3. Code Optimization: Unnecessary duplication in the codebase that could be simplified using helper functions.

Actions Taken:

- Exception Handling: Standardized the exception handling by introducing a base class for exceptions and consistent methods for their management across all modules.
- Security Improvements: Added checks and validation for script deletion functions to prevent unauthorized access and ensure data integrity.
- Code Refactoring: Common functionalities were abstracted into utility functions to reduce redundancy and improve maintainability.

Code Changes:

- Added a new base exception class.
- Implemented new validation logic for script deletions.
- Refactored redundant code patterns into shared utility functions.

Unchanged Aspects:

- Some suggested optimizations were deferred to the next iteration due to time constraints.

Code Review for Second Version

Selection Criteria:

- Code associated with newly introduced features and critical bug fixes.
- Modules where significant refactoring was done in the previous iterations.

Process:

- Reviewed changes made in response to feedback from the first review.
- Used a shared Git repository to highlight changes, facilitating easier discussion and annotation.

Feedback Received:

1. Data Handling: Concerns over handling large data sets with the new script functionalities.
2. Error Handling: Need for more detailed client-side error messages for better user experience.
3. Performance Issues: Potential slow-downs due to inefficient database queries in script management.

Actions Taken:

- Data Handling: Implemented pagination and optimized queries to handle large data sets efficiently.
- Error Handling: Enhanced the granularity of error responses to provide more specific feedback to the client.
- Performance Optimization: Refactored database access layers to improve performance and scalability.

Code Changes:

- Introduced pagination for listing scripts.
- Enhanced error handling mechanisms.
- Optimized database queries to improve response times.

Unchanged Aspects:

- Certain error handling improvements were postponed to future reviews for further assessment and testing.

Code Review for Script List

Project Name/Module Name: Execution Management

Author: @yijun liu

Reviewer: @Junhao KONG

Date: Apr 27, 2024 May 31, 2024

Tools Used: Manual inspection, VS Code for syntax highlighting.

Code Review for the First Version of Code

General Observations

- The code lacks concise comments, which can make it hard for others to understand the purpose and functionality of each section or function.
- There are some commented-out code blocks (e.g., buttons for importing and exporting) that either should be removed or clearly explained why they are retained.

Specific Issues and Suggestions

1. React Component Structure

Issue: The usage of inline functions in JSX props can cause performance issues due to unnecessary re-renders. **Suggestion:** Convert these inline functions to named functions outside the component body or use `useCallback` to wrap these functions, which can prevent them from being recreated on every render.

2. State Management

Issue: The management of state and side-effects can be optimized. The functions and state updates seem scattered and could be more organized. **Suggestion:** Consider using `useReducer` for complex state logic or moving related state handling into custom hooks to clean up the main component body.

3. Accessibility and Usability

Issue: Same as previously identified, input fields lack proper accessibility features. **Suggestion:** Add `aria-label` attributes to input fields and buttons to improve accessibility. Ensure all interactive elements are accessible via keyboard navigation.

4. Error Handling

Issue: There is a lack of error handling around state updates and asynchronous operations. **Suggestion:** Implement robust error handling, especially in asynchronous functions like `handleAddScrip`. Use try/catch blocks and provide user feedback on errors.

5. File Handling

Issue: Icons are imported from SSR-specific files, which might not be optimal for client-side rendering. **Suggestion:** Change the imports for icons from SSR to client-specific modules to avoid potential rendering issues on the client side.

Code Review for the Second Version of Code

General Observations

- Better organization of code with proper modularization of functions and separation of concerns.

- Usage of hooks like `useEffect` for fetching data and managing side effects shows good practice in managing lifecycle events in functional components.

Specific Issues and Suggestions

1. Security and Validation

Issue: Direct execution of scripts or commands without proper validation or sanitization. **Suggestion:** Ensure all file and command executions are properly sanitized to prevent security vulnerabilities such as SQL injection or XSS. Validate all inputs both on the client and server sides.

2. Performance Optimization

Issue: Inline function definitions still persist in the second version, which can lead to the same performance issues. **Suggestion:** Use `useCallback` for all handlers and define these handlers outside of the component or memoize them to prevent unnecessary re-renders.

3. Error Handling and Feedback

Issue: The current implementation does not adequately handle errors or provide feedback for operations like deleting or favoriting scripts. **Suggestion:** Implement more comprehensive error handling and provide clear user feedback for operations. Use mechanisms like toasts or dialog boxes to inform the user of the status of their actions.

4.Accessibility Improvements

Issue: Continued lack of comprehensive accessibility features. **Suggestion:** Expand the use of accessibility features beyond `aria-labels`. Consider implementing keyboard navigation support and focus management to enhance usability for all users.

Next Steps

- **Enhance Security Measures:** Add more robust validation and sanitization for file handling and command execution.
- **Optimize Handler Functions:** Continue refactoring handlers using `useCallback` to improve performance..

Code Review for Log-in, API connection, Web UI integration

Project Name/Module Name: Log-in, API connection, Web UI integration

Author: @Chloe_Duan

Reviewer: @Zhihao Liang

Date: Apr 23, 2024 Apr 27, 2024 May 1, 2024

Tools Used: Manual inspection, VS Code for syntax highlighting.

1. Log-in Component Review

Code Snippet for Review:

```
  const onSubmitRefactor = React.useCallback(
  async (values: Values): Promise<void> => {
    setIsPending(true);
    try {
      const result = await authClient.signInWithPassword(values);

      // If error happens when connecting API
      if ('error' in result) {
        setError('root', { type: 'server', message: result.error });
        setIsPending(false);
        return;
      }

      await checkSession?.();

      // UserProvider, for this case, will not refresh the router
      // After refresh, GuestGuard will handle the redirect
      router.refresh();
    } catch (error) {
      console.error('Error during login:', error);
      setError('root', { type: 'server', message: 'An unexpected error occurred' });
    } finally {
      setIsPending(false);
    }
    router.refresh();
  },
  [checkSession, router, setError]
);

async signInWithPassword(params: SignInWithPasswordParams): Promise<ApiStatusResponse> {
  try {
    const response = await login(params);

    if (!response.success) {
      return { error: response.message || 'Login failed' };
    }

    // console.log('Login successful:', response.data);
    if (!response.data.user) {
      return { error: 'User data is missing' };
    }
```

```
      const user = parseUser(response.data);
      localStorage.setItem('custom-auth-token', response.data.access_token);
      return user;
    } catch (error) {
      console.error('Error during signInWithPassword:', error);
      return { error: 'Email or password is incorrect.' };
    }
```

Review Points:

- **Logical Errors:** Check if the condition to determine when to send the request is adequate. Should we check for just empty strings, or also null/undefined values?
- **Code Standard Compliance:** Ensure naming conventions are followed and that the code adheres to ES6+ standards if applicable. Is the fetch API used correctly?
- **Code Simplicity:** Is the error handling strategy adequate? Could it be simplified or made more robust?
- **Reusability:** The function could be more reusable if it accepted a callback or returned a promise that resolves or rejects based on the authentication success.

Next Steps After Review:

- **Pull Request:** Incorporate any accepted changes based on feedback and prepare for a merge after final approval.
- **Feedback Integration:** Address all comments critically, improving error handling, perhaps by adding specific error messages for the user based on different failed states (invalid credentials, server error, etc.).

2. API Connection Component Review

Code Snippet for Review:

```
  export interface ListUsersResponse {
  avatar?: string;
  created_at?: string;
  created_by?: string;
  email: string;
  id: number;
  last_login?: string;
  role_id: number;
  updated_at?: string;
  username: string;
}

// list users
export async function listUsers(): Promise<APIResponse<ListUsersResponse[]>> {
  return request<ListUsersResponse[]>({
    url: '/api/users',
    method: 'GET',
  });
}
  useEffect(() => {
    const loadData = async () => {
      await fetchUsers();
    };
    toast.loading('Loading users...');
    loadData()
      .then(() => {
        toast.success('Users loaded successfully');
      })
      .catch((e) => {
        console.log(e);
```

```
      })
      .finally(() => {
        toast.dismiss();
      });
  }, []);

  const fetchUsers = async () => {
    try {
      const response = await listUsers();
      setUsers(response.data.map(parseUser));
    } catch (error) {
      console.error('Failed to fetch users:', error);
    }
  };
```

Review Points:

- **Logical Errors:** Ensure proper error handling is implemented.
- **Code Standard Compliance:** Is async/await used properly? Are there any potential unhandled promise rejections?
- **Code Simplicity:** Is the function handling more than it should? Could the error handling be more specific based on the error codes?
- **Reusability:** This function is quite reusable; however, parameterizing the error message might make it more versatile.

Next Steps After Review:

- **Pull Request:** Update the function to handle different types of HTTP status codes appropriately.
- **Feedback Integration:** Modify the function based on received feedback to handle specific HTTP errors and possibly retry logic for specific failures.

3. Web UI Integration Component Review

Code Snippet for Review:

```
import { getSiteURL } from '@/lib/get-site-url';
import { LogLevel } from '@/lib/logger';

export interface Config {
  site: { name: string; description: string; themeColor: string; url: string };
  logLevel: keyof typeof LogLevel;
}

export const config: Config = {
  site: { name: 'Py Runner', description: '', themeColor: '#090a0b', url: getSiteURL() },
  logLevel: (process.env.NEXT_PUBLIC_LOG_LEVEL as keyof typeof LogLevel) ?? LogLevel.ALL,
};

import type { NavItemConfig } from '@/types/nav';
import { paths } from '@/paths';

export const navAdminItems = [
  { key: 'user', title: 'User Management', href: paths.dashboard.customers},
] satisfies NavItemConfig[];

export const navItems = [
  { key: 'overview', title: 'Overview', href: paths.dashboard.overview},
  { key: 'script', title: 'Script List', href: paths.dashboard.scripts},
  { key: 'account', title: 'Account', href: paths.dashboard.account},

] satisfies NavItemConfig[];
```

Review Points:

- **Logical Errors:** Check if DOM elements exist before adding event listeners.
- **Code Standard Compliance:** Use modern DOM manipulation techniques if applicable.
- **Code Simplicity:** Could these methods be simplified or broken down into smaller parts?
- **Reusability:** Is there a way to make this component more modular?

Next Steps After Review:

- **Pull Request:** Refactor the UI class to use more modular methods and modern JavaScript features.
- **Feedback Integration:** Adjust based on feedback, possibly changing how DOM elements are managed or introducing a state management system.

In each case, after incorporating the feedback, further tests should be conducted to ensure that the changes are effective and do not introduce new issues. Depending on the feedback's nature, not all suggestions might be implemented, particularly if they conflict with other design considerations or project constraints.

Code Review for favourite API

Module Name: Favorite function

Author: Lingyi Kong @Lingyi Kong

Reviewer: Yuncong Ji @Yuncong Ji

Review Date: 23/4

Code Review

1. Code Structure and Naming Convention:

- [x] Code structure appears clear and organized.
- [ ] Function and variable names are descriptive and adhere to the naming conventions.

2. Functionality Implementation:

- [x] `add_favorite`: Validates if the script is already in favorites or if the script exists before adding it.
- [x] `delete_favorite`: Properly handles the case where the favorite to be deleted doesn't exist.
- [x] `list_favorite`: Checks if the user exists before listing favorites.

3. Comments and Documentation:

- [x] Add comments/docstrings explaining the purpose of functions and any complex logic, especially in utility functions like `add_favorite`, `delete_favorite`, and `list_favorite`.

4. Test Coverage:

- [ ] Include unit tests to cover different scenarios for `add_favorite`, `delete_favorite`, and `list_favorite` functions.

5. Performance and Security:

- [ ] Ensure that sensitive operations like adding/deleting favorites are protected with JWT authentication.

6. Maintainability:

- [x] Consolidate database session management into a reusable function to reduce code duplication.
- [ ] Use constants for HTTP status codes and error messages to enhance maintainability and consistency.
- [ ] Evaluate potential improvements for error handling, such as custom exceptions for clearer error propagation.

Next review action: 28/4

Additional Suggestions:

- Consider adding validation for `script_id` parameter in API endpoints to ensure it's a positive integer.
- Refactor repetitive code, such as database session management, into utility functions or decorators for cleaner codebase maintenance.

Module Name: Favorite function

Author: Lingyi Kong @Lingyi Kong

Reviewer: Yuncong Ji @Yuncong Ji

Review Date: 28/4

Code Review

1. Code Structure and Naming Convention:

- [x] Code structure appears clear and organized.
- [x] Function and variable names are descriptive and adhere to the naming conventions.

2. Functionality Implementation:

- [x] `add_favorite`: Validates if the script is already in favorites or if the script exists before adding it.
- [x] `delete_favorite`: Properly handles the case where the favorite to be deleted doesn't exist.
- [x] `list_favorite`: Checks if the user exists before listing favorites.

3. Comments and Documentation:

- [x] Add comments/docstrings explaining the purpose of functions and any complex logic, especially in utility functions like `add_favorite`, `delete_favorite`, and `list_favorite`.

4. Test Coverage:

- [x] Include unit tests to cover different scenarios for `add_favorite`, `delete_favorite`, and `list_favorite` functions.

5. Performance and Security:

- [x] Ensure that sensitive operations like adding/deleting favorites are protected with JWT authentication.

6. Maintainability:

- [x] Consolidate database session management into a reusable function to reduce code duplication.
- [x] Use constants for HTTP status codes and error messages to enhance maintainability and consistency.
- [x] Evaluate potential improvements for error handling, such as custom exceptions for clearer error propagation.